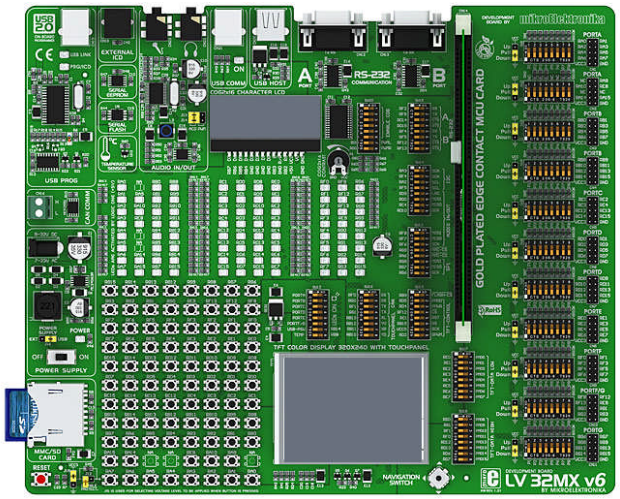
型番:PICD-32MX2　　REV.1.00

USB2.0対応 PICライター搭載
高機能PIC32MX開発用統合評価ボード

取扱説明書
お使いになる前にこの説明書をよくお読みの上正しくお使いください。

(C)2010 マイクロテクニカ

ボード本体



製品の概要

高機能PIC32MX開発用統合評価ボード[型番:PICD-32MX2、以下PICD-32MX2と記載]は、オンボードにUSB接続のPICライターを搭載し、最新の32ビットPIC、PIC32MXの開発に対応した統合開発ボードです。

ボード上には、PIC32MXシリーズのデバイスを実装したMCUカードが装着できるSIMMスロットが搭載されており、標準ではPIC32MX460F512Lを実装したMCUカードが付属しています。

ボード上に搭載のPICライターは、最新のPIC32MXシリーズのデバイスに対応しており、基板上に装着したMCUカードに搭載のマイコンにプログラムを書き込みます。USB接続を行った場合、USBバスからPICD-32MXの電源を取りますので、別途電源を準備する必要がなく、パソコンと接続すればすぐに開発を始められます。

ボードに搭載されている各周辺回路に接続されているピンは周辺回路に接続して利用するか又は周辺回路には接続しないで使用するかをディップスイッチやジャンパーソケットで選択できます。またI/Oピンはボード右側に実装されているヘッダピンから取り出すことができます。

周辺回路やI/Oピンの状態を表示するLEDやタクトスイッチ等を使うことでPIC32MXの評価からアプリケーションの開発、実験が簡単に行えます。

マイクロチップ社からの純正Cコンパイラー、MPLAB C32を利用することでCコンパイラーによる開発が可能となります。32ビットPICのPIC32MXの開発を簡単にはじめたい方におすすめの評価ボードです。

パッケージの内容

■同梱物
- ・PICD-32MX2ボード本体
- ・PIC32MX460F512L-80搭載MPUボード
- ・USBケーブル
- ・CD-ROM
- ・マニュアル(本書)

ドライバのインストール及びパソコンとの接続

■デバイスドライバーのインストール

PICD-32MXをパソコンと接続する前に、デバイスドライバーをパソコンにインストールします。(Windows98及びMEをご使用のお客様は本項は読み飛ばして、次の「パソコンと接続する」の項よりお読み下さい。)WindowsVista、7及びWindowsXPをご利用のお客様は下記の手順で、ドライバーをインストールします。

1 付属のCD-ROMをパソコンのCD-ROMドライブに挿入してエクスプローラー等で内容を表示します。

2 "USB Programmer"のフォルダを開きます。

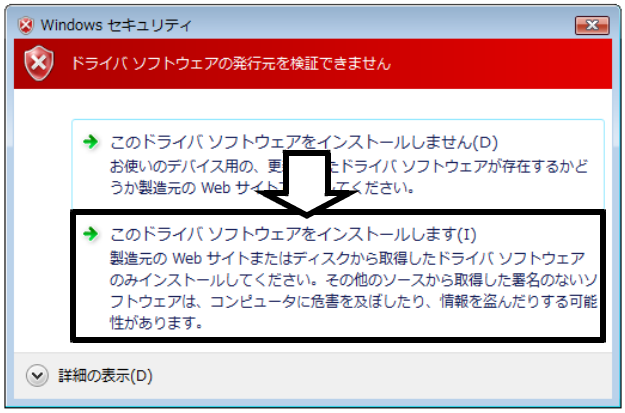
3 "Driver"フォルダを開きます。

4 OSごとにフォルダがあります。ご使用のパソコンのOSにあったフォルダを開きます。(WindowsVista,7用には32ビット用と64ビット用がありますのでご注意ください。)

5 開くと実行ファイル、"USB18PRG～.exe"がありますのでダブルクリックして実行します。

6 実行するとダイアログが表示されますので、そのまま"次へ"をクリックして続行します。

7 ドライバーのインストールが開始されます。
WindowsVista環境で下記のような警告ダイアログが表示された場合には、"このドライバソフトウエアをインストールします"をクリックして続行してください。

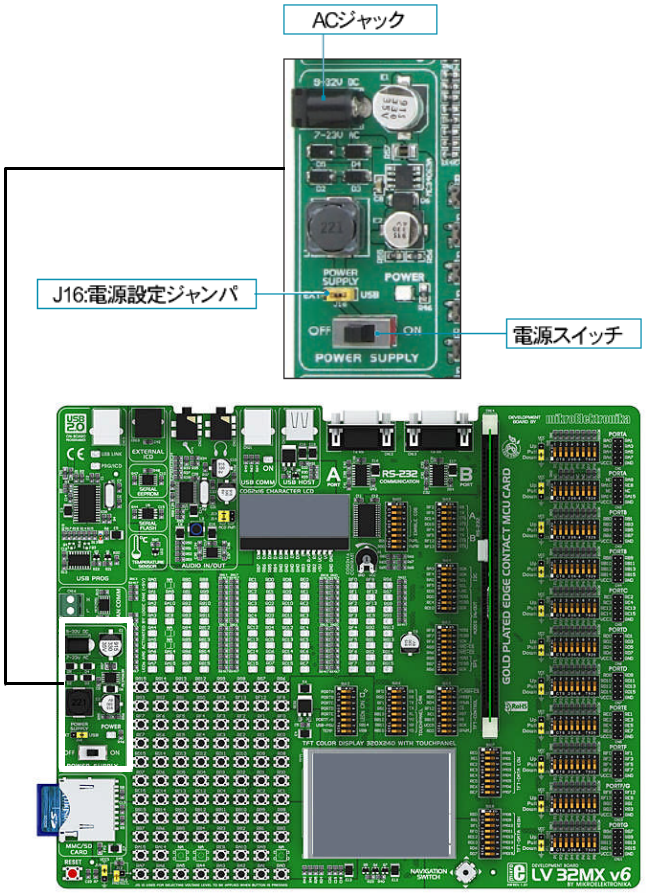


下図のようなメッセージが表示された場合には、"インストール"ボタンを押してください。

8 インストール作業が完了すると、下図のような画面になりますので、"完了"ボタンを押してインストールを完了します。
この時、"状態"のボックスに「使用できます」と表示されていることをご確認下さい。

■パソコンと接続する

1 PICD-32MXの電源スイッチをONにします。
また、PICD-32MXの電源をUSBのバスパワーより給電しますので、J16の"USB"側にソケットを装着してください。



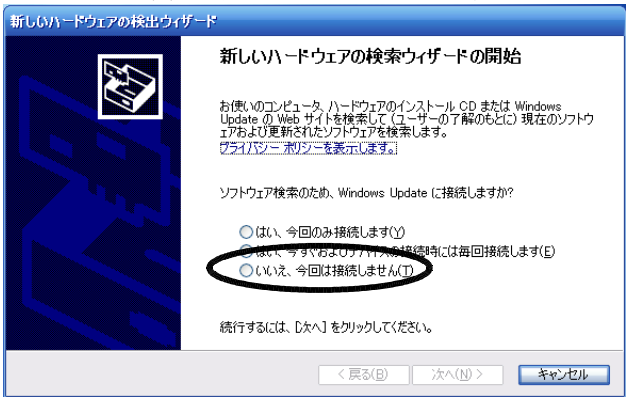
2 USBケーブルで、パソコンのUSBポートと、PICD-32MX2を接続します。

→接続するとPOWERの緑LEDが点灯します。
→新しいハードウエアの検出ウィザードが自動的に起動します。

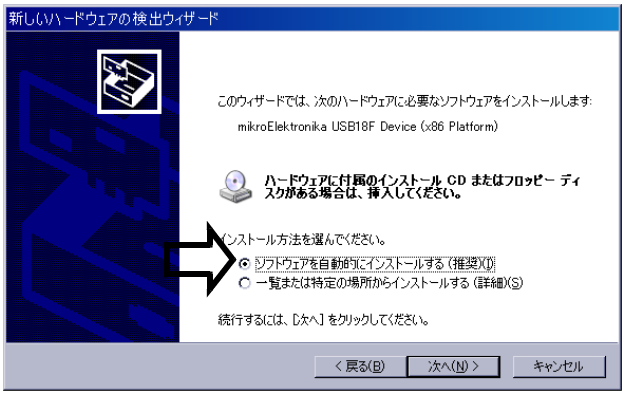
以下OSによって操作方法が異なります。

■Windows2000, XPをご利用の場合----------

3 下図のようなダイアログが表示されたら"いいえ、今回は接続しません"にチェックを入れて"次へ"ボタンをクリックします。



4 下記のようなダイアログが表示されますので、"ソフトウエアを自動的にインストールする(推奨)"にチェックをいれて、"次へ"ボタンをクリックします。



5 「ソフトウエアをインストールしています。お待ち下さい」というメッセージが表示されます。しばらくするとインストールが完了して下図のような表示となりますので"OK"ボタンをクリックして終了します。

■WindowsVista,7をご利用の場合----------

3 WindowsVista,7の場合には、PICD-32MX2を接続すると自動的にデバイスドライバーのセットアップが開始されます。
あらかじめデバイスドライバーは先の手順にてインストールされていますので、WindowsVista,7は自動的にデバイスドライバーを検出してインストールを完了します。

接続後しばらく待ちますと、下図のようなポップアップがタスクバーに表示されます。

上図のホップアップが表示されればインストールは完了です。

■Windows98(SE), MEをご利用の場合----------

3 自動的にデバイスが検出されます。
"新しいハードウエアの追加ウィザード"が表示されたら「次へ」を押して続行します。

4 次のダイアログでは,"使用中のデバイスに最適なドライバを検索する"にチェックを入れて,「次へ」をクリックします。

5 "検索場所の指定"にチェックを入れて,「参照」ボタンを押します。
ドライバの場所を指定するダイアログが表示されますのでCD-ROM内の"USB Programmer Software"フォルダ内の"Driver"フォルダにある"WIN98_ME"フォルダを指定してください。CD-ROMドライブがQドライブの場合にはディレクトリは下記の通りとなります。
Q:¥USB Programmer Softeare¥Driver¥WIN98_ME

指定したら「次へ」を押して続行します。
インストールが完了したら,「完了」ボタンを押して終了します。

～ここからは各OS共通の項目です～

PICD-32MX2ボード上の"USB Link"の黄LEDが点灯していることを確認してください。
黄LEDの点灯はPC側にPICD-32MX2が正しく認識されていることを示します。

※"USB Link"のLEDが点灯していない場合には、正しくドライバーがインストールされていません。再度手順を確認の上、ドライバーをインストールし直しておためしください。

※デバイスマネージャーに"USB Devices by mikroElektronika"が追加されます。

WIndowsVista及び7環境でご使用のお客様へ
ユーザーアカウント制御無効設定のお願い

WindowsVista及び7環境で、PICD-32MX2をご使用の場合、Windowsに搭載されたユーザーアカウント制御(UAC)機能がUSBポートへのアクセスを拒否することにより、下図のようなエラーメッセージが表示されて、書き込みができないという現象を確認しています。

この現象は、WindowsVista及び7において標準で有効になっているユーザーアカウント制御機能を無効に設定することで回避できます。下記の方法で設定を変更して頂けますようお願い致します。

※なおこのエラーは、WindowsVista及び7以外の環境でお使いのお客様には関係ありません。

■Windows Vistaのユーザーアカウント制御を無効に設定する

1 スタートボタンをクリックして、コントロールパネルを表示します。

2 コントロールパネルから"ユーザーアカウント"をダブルクリックします。"ユーザーアカウント"アイコンがない場合には、ウインドウ左上の"クラシック表示"をクリックして表示を変更します。

3 "ユーザーアカウント制御の有効化または無効化"のリンクをクリックします。

4 "ユーザーアカウント制御(UAC)を使ってコンピューターの保護に役立たせる"のチェックを外します。

5 "OK"ボタンを押して完了します。パソコンを再起動します。

■Windows7のユーザーアカウント制御を無効に設定する

1 スタートボタンをクリックして、コントロールパネルを表示します。

2 コントロールパネルから""ユーザーアカウントと家族のための安全設定"をクリックします。

3 "ユーザーアカウント"をクリックします。

4 "ユーザーアカウント制御設定の変更"をクリックします。

5 画面に表示されるスライドバーを一番下の"通知しない"に設定します。

6 "OK"ボタンを押して完了します。パソコンを再起動します。

MPLAB C32のダウンロード

PIC32用のCコンパイラーは、MPLAB C32というコンパイラーが、マイクロチップ社からリリースされています。製品版も販売しておりますが、Evaluation Versionという無償で利用できるものがあります。

Evaluation Versionは、最初の60日間は製品と同等の内容で動作しますが、期限が超過すると、最適化機能が固定されとなり、ファイルサイズが多少大きくなったり、実行速度が遅くなったりします。

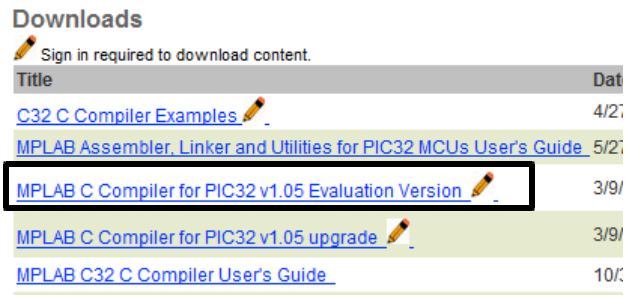
ただし、その他の機能は製品版と全く同じで、大規模なプログラムでなければ実質このEvaluation Versionで十分開発が進められます。

MPLAB C32は、MPLABという統合開発環境の中で動作します。MPLABは付属のCD-ROMに収録されていますが、MPLAB C32はマイクロチップ社のWEBサイトにユーザー登録してからダウンロードすることになります。下記の手順でダウンロードしてください。

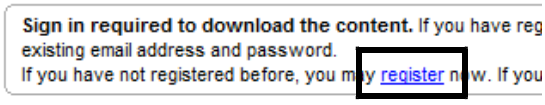
1 当方のダウンロードページにアクセスします。
http://www.microtechnica.net/manual/

2 項目20番にMPLAB C32のダウンロードサイトへのリンクがありますので、リンクをクリックします。マイクロチップ社のMPLAB C32のページが表示されます。

3 画面下の方に下がると"Download"という項目がありますので、その一覧から"MPLAB C Compiler for PIC32 v1.xx Evaluation Version"というリンクをクリックします。



4 "Sing In"画面が表示されます。マイクロチップ社のアカウントをお持ちでしたらメールアドレスとパスワードを入力して"Sing In"をクリックします。
もしアカウントをお持ちでない場合には、新規登録をします。画面中央上部にある"register"というリンクをクリックします。



5 画面の指示に従い必要事項を入力します。
登録が正しく完了すると、入力したメールアドレスとパスワードによってサインインできるようになりますので、サイトにサインインします。

6 手順3の画面を表示して、"MPLAB C Compiler for PIC32 v1.xx Evaluation Version"というリンクをクリックし、ソフトウエアをダウンロードしてください。

ソフトウエアのインストール

開発に必要なソフトウエアをインストールします。なお、ソフトウエアをインストールする際には、必ずUSBポートからPICD-32MX2を外した状態で行ってください。

■MPLABのインストール

MPLABはPICマイコンのプログラム開発用の統合開発環境です。PICマイコンの開発元マイクロチップ社から提供されています。MPLABがお手持ちのパソコンにインストールされていない場合には、インストールしてください。

1 CD-ROM内の"MPLAB"フォルダを開きます。
"SETUP.EXE"をダブルクリックして実行します。
"Next>"をクリックして続行します。

2 "License Agreement"のダイアログが表示されたら内容をよくご確認の上、同意する場合には"I accept the terms of the licence agreement"にチェックを入れて、"Next>"クリックして続行します。
なお同意しないとインストールは続行できません。

3 "setup type"の選択ダイアログではすべてのプログラムをインストールしますので"Complete"にチェックを入れて"Next>"をクリックします。

4 MPLABをインストールするディレクトリを指定します。
特に設定する必要がなければデフォルトのままにしておきます。
インストールするディレクトリを変更したい場合には,"Browse"ボタンを押してインストールディレクトリを設定します。
なお変更した場合には,インストールディレクトリをメモするなど忘れないようにしてください。
"Next>"をクリックして続行します。

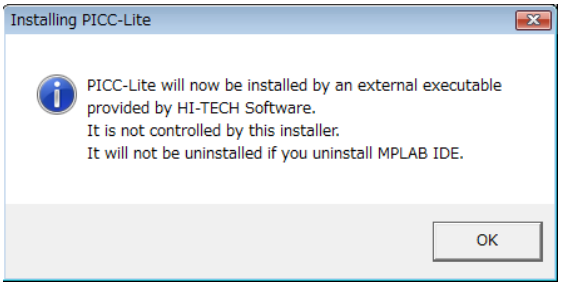
5 以降のダイアログでは次のように進めます。

■Application Maestro License
→"I accept the terms of the license agreement"にチェックをいれて"Next>"をクリックします。

■MPLAB C32 License
→"I accept the terms of the license agreement"にチェックをいれて"Next>"をクリックします。

■Start copying files
→"Next>"を押します。
・・・インストールが開始されます。完了までしばらく待ちます。

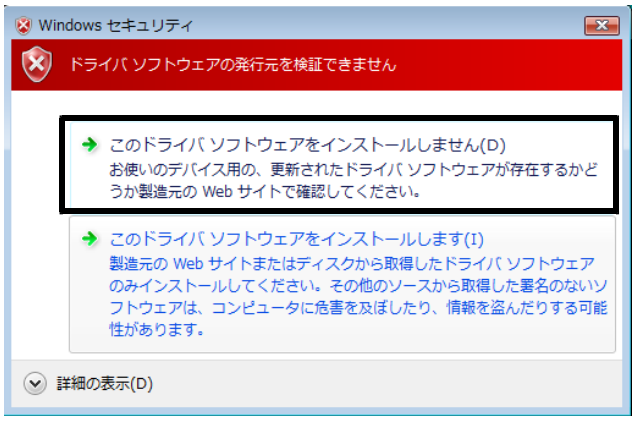
6 インストールが完了すると、下図のようなダイアログが表示されます。MPLABバージョン8.00以降からはPICCコンパイラーのライトバージョンがMPLABに付属するようになったためです。
ここではOKボタンを押して続行します。



7 下図のようなPICC-Liteのインストール画面が表示されますので、ここでは"Cancel"ボタンを押してインストールをキャンセルします。

※PICC-Liteは、Hi-Tech社のPIC用Cコンパイラーです。インストールしていただいてもかまいませんが、本キットではこのソフトウエアは使用しません。

8 Windows Vista環境にインストールされている場合には、最後に下図のようなドライバーインストールに関するセキュリティ警告が表示されることがあります。
この場合には、"このドライバソフトウエアをインストールしません(D)"をクリックしてください。
ダイアログが消えたら"Finishボタンを押してインストールを完了します。



■MPLAB C32のインストール

先ほどダウンロードしたPIC32用のCコンパイラー、MPLAB C32をインストールします。

1 ダウンロードした実行ファイルをダブルクリックして実行します。
インストーラーが起動します。

2 ウィザードが起動したら"Next"をクリックして続行します。
"License Agreement"が表示されますので、同意される場合には"I accept the terms of the license agreement"にチェックを入れて、"Next"ボタンを押します。

3 "License Key"の入力する画面となりますが、ここでは空欄のまま"Next"を押して続行します。

4 インストールディレクトリを指定します。このとき、必ず先にインストールしたMPLABがインストールされているディレクトリと同じ"Microchip"ディレクトリの下にインストールするよう設定してください。

通常はそのままでかまいませんが、MPLABをインストールしたとき、インストールディレクトリをデフォルト設定から変えていたり、Cドライブ以外のドライブにインストールした場合には、ここで設定を変更する必要があります。間違えないようご注意ください。

5 "Install"ボタンをクリックするとインストールが始まります。
画面の指示に従って、完了します。

■PICプログラマ用のソフトウエア

PICへプログラムを書き込む際に使用するプログラマソフトウエアです。下記の手順でインストールを行います。

1 CD-ROM内の"USB Programmer Software"内にある"SETUP.EXE"をダブルクリックして実行します。インストーラーが起動しますので、"Next"をクリックして続行します。

2 ライセンス許諾画面が表示されますので、同意される場合には、"I accept the terms in the License Agreement"の方にチェックを入れて、"Next"ボタンをクリックします。

3 次の画面はそのまま"Next"をクリックします。

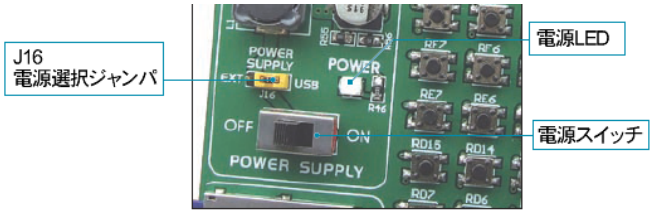
4 インストールするディレクトリを指定します。
"Install"ボタンを押すとインストールが開始されます。

5 インストールが完了したら"Finish"ボタンを押して完了します。

電源の投入方法

PICD-32MX2は、USB接続にてUSBバスから電源の供給を受けることができます。この場合、別途電源を供給する必要はありません。しかしPICD-32MX2側で多くの電流(およそ350mA以上)を消費する予定がある場合にはUSBバスパワー給電では電力が不足する場合があります。この場合には別途外部からACアダプタにて電源を供給する必要があります。

給電方法はジャンパーソケットJ16によって設定を行います。
なお工場出荷時の設定はUSBバスパワー給電の設定になっています。



■USBバスパワーから給電する場合

PICD-32MX2ボード上の"SUPPLY SELECT"ジャンパーピンを「USB」側にショートします。

■ACアダプタから給電する場合

PICD-32MX2ボード上の"SUPPLY SELECT"ジャンパーピンを「EXT」側にショートします。

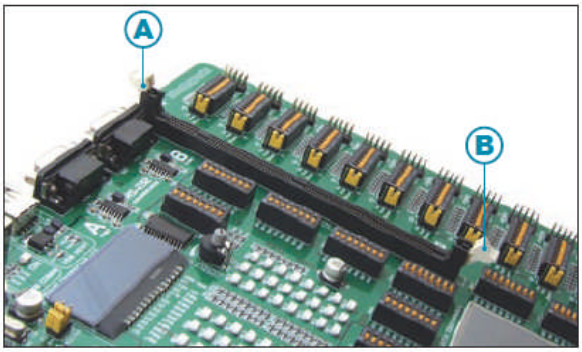
※ACアダプタは,DC9V～12Vで500mA以上のものをご用意ください。径は2.1φです。(当方にて専用ACアダプタを販売中です)

MCUカードの装着

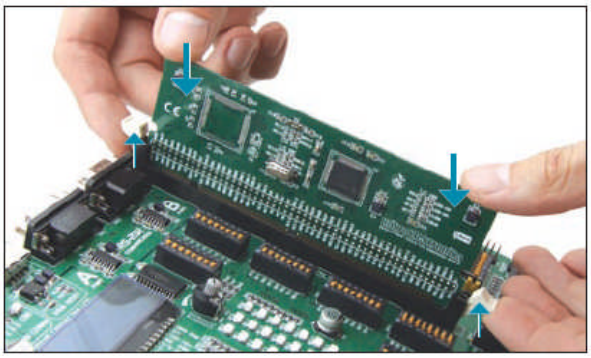
お買い上げ時は、PIC32MX460F512L−80を搭載したMCUカードが付属しています。使用する前に、このMCUカードを正しくボードに装着する必要があります。下記の手順で慎重に行ってください。

1 PICD−32MX2の電源を切断します。

2 DIMMスロットの両端にある白いレバーを左右同時に下に倒します
※下図のAとBのレバー

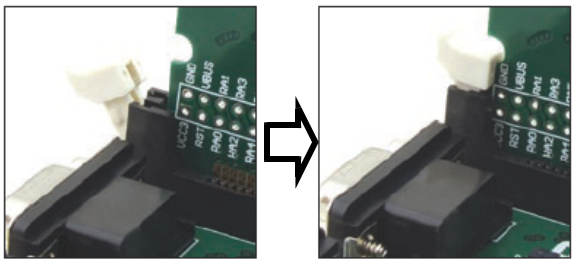


3 MCUカードを、DIMMスロットとMCUカードの切り欠きの位置を合わせて、真っ直ぐ下に押し込んで下さい。



斜めに押し込むとスロットが破損することがあります。
真っ直ぐとDIMMスロットの溝に沿うようして下に押し込みます。

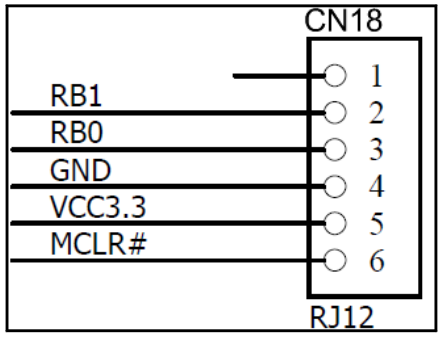
4 押し込みながら、DIMMスロットの白いレバーを起こします。
白いレバーがMCUカードの切り欠き部分に入り込み、カードが固定されます。



正しく固定されてMCUカードがグラグラと動かないことを確かめます。
外す際には、DIMMスロットのレバーを倒すとMCUカードを外すことができます。

MPLAB-ICDを使用する場合

PICD−32MX2には、マイクロチップ社純正のICD、MPLAB−ICD2又は3を接続するための6ピンモジュラーコネクターが付いています。ここにMPLAB−ICDを接続することでPICD−32MX2をターゲットボードとして使用することができます。配線は下図の通りです。



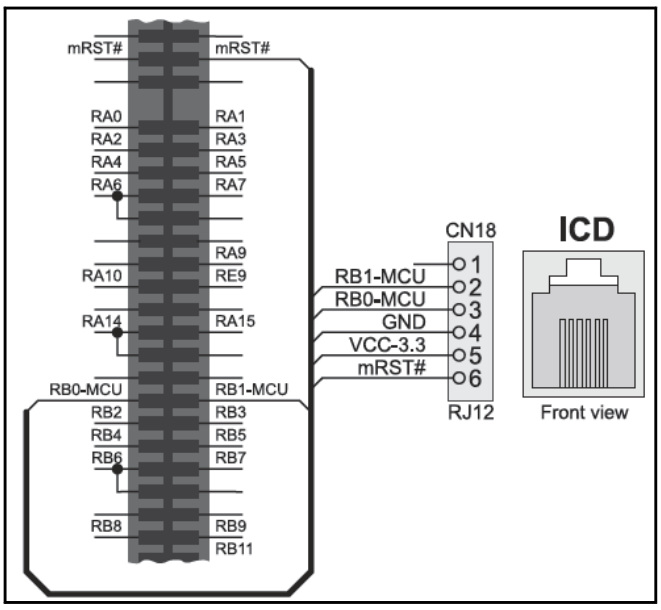
MPLAB−ICDを利用する場合には、RB1及びRB0はMPLAB−ICD側に占有されますので外部回路を接続したり、プルアップやプルダウンをしないようにします。

■MPLAB−ICD使用時のPICD−32MX2の電源について

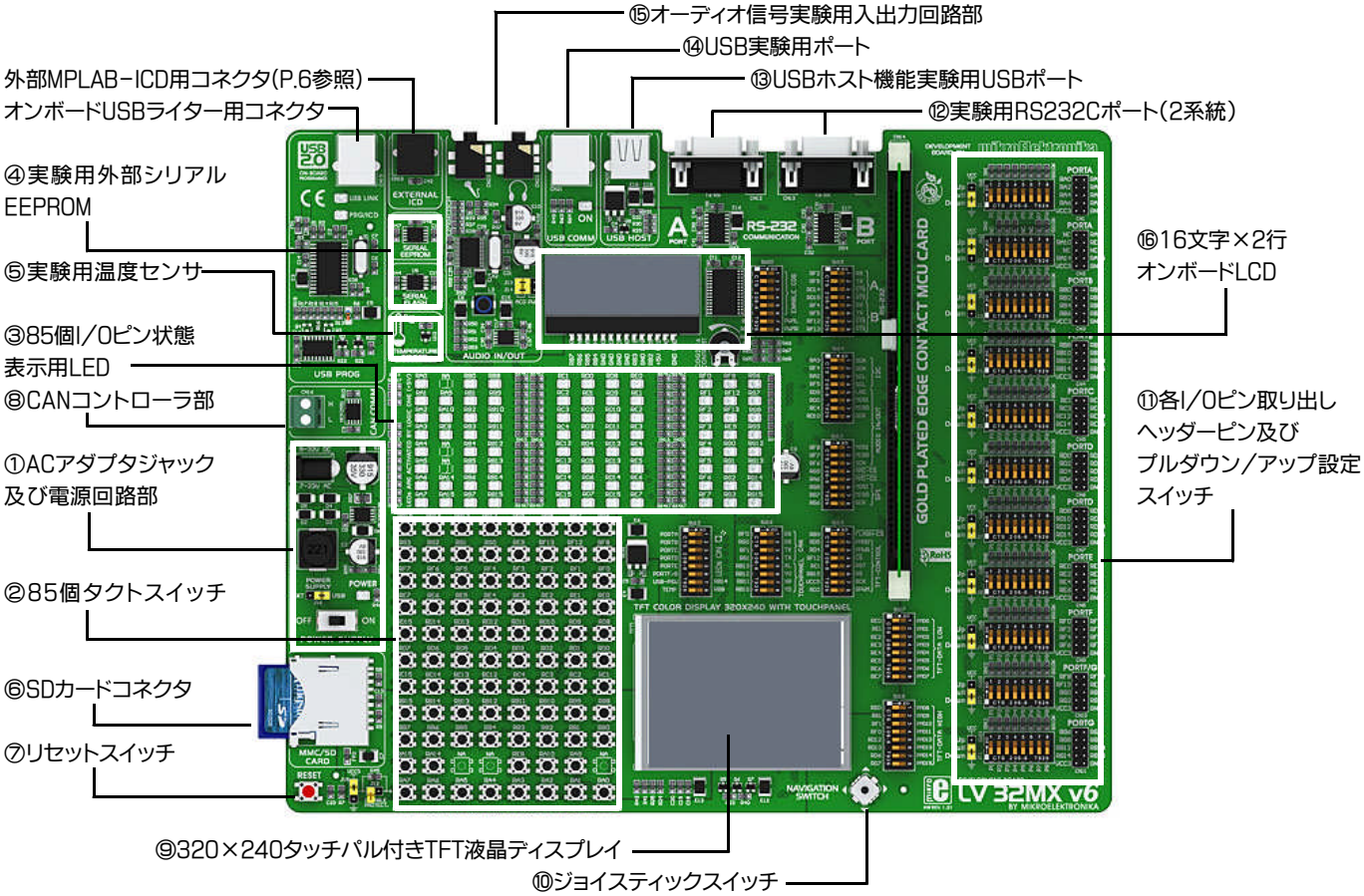
MPLAB−ICD用のモジュラーコネクタのVddピンは、PICD−32MX2のVdd(+3.3V)と接続されています。MPLAB−ICD接続時には、電源の取り扱い及びMPLAB−ICDの電源設定を下記のように設定してください。設定を間違うと、機器を破損させる恐れがありますのでMPLAB−ICDを接続する前に必ず配線や設定が正しいか十分ご確認の上接続をしてください。

・PICD−32MX2の電源は必ずACアダプタにて外部電源から給電する
・PICD−32MX2のUSBはパソコンと接続しない
・MPLAB−ICDは、"Target has own power supply"に設定する
→ターゲットボードの自己給電

以上の設定を正しく行ってご使用ください。MPLAB−ICDは、ターゲットボードに対してMPLAB−ICD側から電源を供給するオプション設定がありますが、この設定を行うと、PICD−32MX2ボードの消費電流が大きいため、MPLAB−ICDが破損します。十分にご注意ください。



PICD-32MX2の各部の説明



①ACアダプタジャック及び電源セレクトジャンパ

PICD-32MX2の電源をACアダプタから給電する場合には、このジャックにDC9V出力で2.1φのACアダプタを接続します。センター極性は問いません。USBバスパワー給電では、およそ350mAまで使用できますがそれ以上の電流を消費するような場合にはACアダプタを使用してください。

給電方法はJ16の"SUPPLY SELECT"ジャンパーソケットで選択します。USBバスパワーから給電する場合はUSBと印刷された方に、ACアダプタから給電する場合はEXTと印刷された方にジャンパーソケットをセットします。

②85個I/Oピン接続タクトスイッチ群

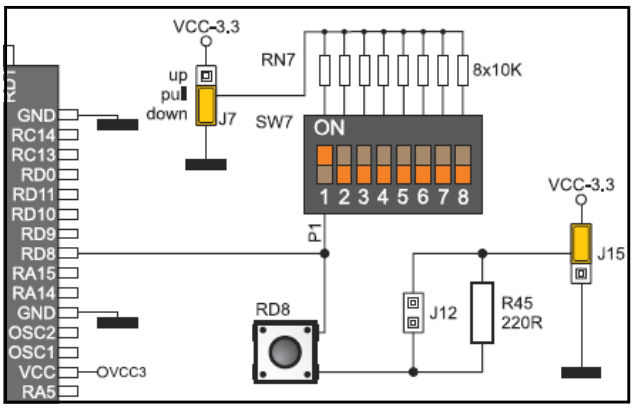
J15のジャンパー設定によって、アクティブHigh又はアクティブLowが選択可能な85個のタクトスイッチです。

タクトスイッチ群の左側にあるJ15をVcc3と刻印された側に設定すると、スイッチを押した時に該当ピンに3.3Vの電圧が印加されるアクティブHighに、GND側に設定すると、スイッチを押した時に該当ピンはGNDと接続され、Lowレベルになりアクティブになります。

各スイッチのプルアップ及びプルダウンは、各ポート8ビット単位(上位8ビット、下位8ビット)で設定できます。プルアップ/プルダウンの設定は、"⑪全I/Oピン取り出しようヘッダーピン"の部分で設定します。ジャンパー設定J1～J11の設定と、その設定を有効又は無効にするディップスイッチSW1～SW11によって設定できます。

タクトスイッチの設定は、J15のアクティブLow/High設定と、J1～J11のプルアップ/プルダウンの設定を適切に組み合わせる必要があります。J15をアクティブHigh設定にした場合(ボタンを押した時H)には、該当するタクトスイッチの接続されたピンは、プルダウン(定常時はL)にしておく必要があります。

下図はRD8の場合の例です。



ジャンパーJ7でプルダウンに設定し、その設定をRD8に対して有効にするため、SW7の1番ピンをON側に設定します。これで定常時はプルダウンされてRD8はLレベルとなります。

タクトスイッチ側は、J15をVcc3側に設定することで、スイッチを押した時Hになります。

これで、RD8はスイッチを押さない時はLレベル、スイッチを押した時にHレベルになる状態になります。

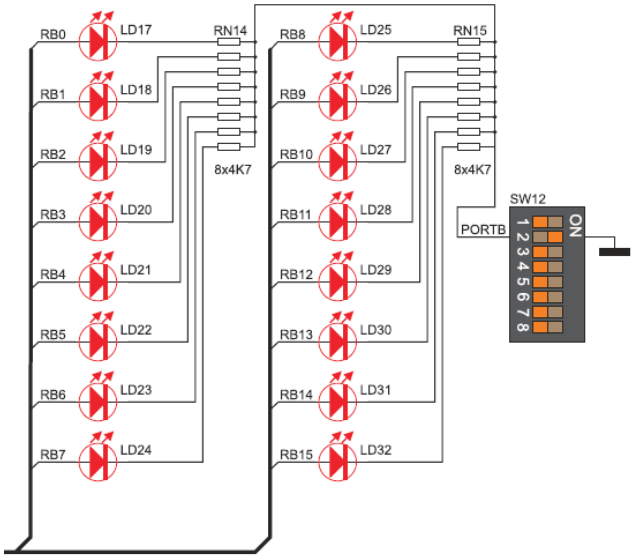
J12は、短絡時の過電流防止用ジャンパーです。ジャンパーをセットしないと、220Ωの抵抗器によってスイッチがVcc又はGNDに接続さ

れます。これによって、例えば通常はPIC側の設定が出力ピンの設定の時はスイッチは押しませんが、誤って該当ピンが出力ピンに設定されており、PIC側がHのロジックだった時、J15がGNDと接続させる設定になっていると大きな電流がPIC側からGNDに流れ、PICの破損につながります。このJ12のジャンパーソケットを外しておくことで、220Ωの抵抗器を介して電流が流れることになりますので、誤った設定による過電流からPICのピンの破損を保護することができます。
通常は、J12はオープン(ジャンパーソケットを装着しない状態)にしておくことが推奨されます。

③85個I/Oピン状態表示LED群

PORTA(RA)～PORTG(RG)の各ビットの状態を表示できるLED群です。RB・RDは16ビット全ビットにLEDが接続されています。その他のポートはユーザーが使用できるビットに対してLEDが接続されています。該当のピンがHレベルになると点灯します。

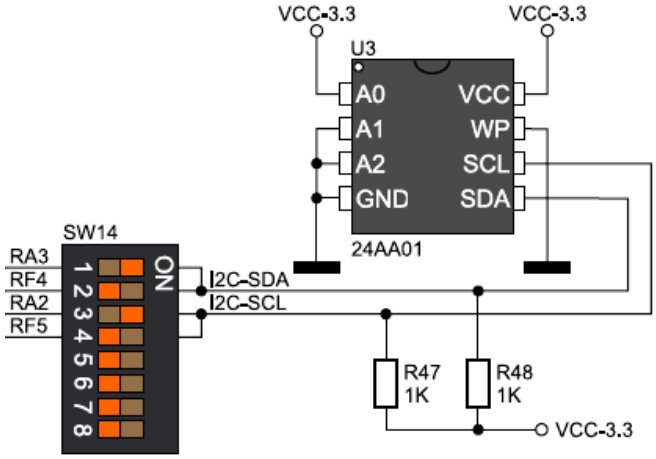
LEDとマイコンのピンを接続するか、接続しないかの設定はディップスイッチSW12で設定できます。
と基板上に印刷された方にポート名のスイッチをON側に設定すると、そのポートに接続されたLEDのカソード側がGNDと接続され、該当ピンがHレベルの時、LED側に電流が流れ、LEDが点灯するようになります。下図はPORTBの接続例です。



マイコンのI/OピンにLEDが物理的に接続されていると、LED側に電流が流れ電圧降下が発生します。よって、ピンをI/Oピンとして使用する場合、LEDと接続されていると電圧降下により、Hレベル時にVcc電圧となりません。H-LにてVp-pをVcc-GND電圧で使用したい場合にはSW12の該当ポートのスイッチをOFF側に設定して、物理的にLEDからI/Oピンを切り離してご利用ください。

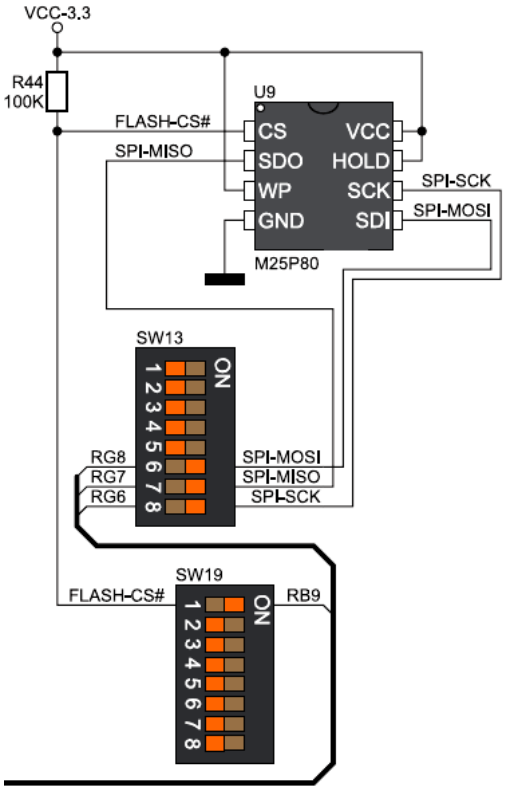
④実験用シリアルEEPROM

実験用及びマルチメディア関連の大きなデータを保存するたに使用できる2つのメモリーが基板に実装されています。
"SERIAL EEPROM"と記載の方には、I2Cインターフェイス式のEEPROM、24AA01(最大1kbitデータ)が接続されています。
I2C通信での通信実験やEEPROMへのデータの書き込み、読み出しの実験に使用できます。
I2CのSDA及びSCLの結線は、ディップスイッチSW14によって、PICの、どのピンと接続するかを指定できます。SDAはRA3又はRF4と、SCLはRA2又はRF5と接続することができます。なお両信号線は、1KΩの抵抗器によって、プルアップされています。



I2Cのスレーブアドレスは0になっています。

"SERIAL FLASH"と記載の方には、SPI通信式の8Mビットフラッシュメモリーの M25P80が搭載されています。主にマルチメディア関連のデータを記憶したりする目的で使用します。もちろんSPI通信やフラッシュメモリーの実験や学習にも使用できます。
信号線のCS、MISO、MOSI、SCKの4線は、SW13及びSW19によってPICと接続するかどうかを指定できます。接続できるピンは決まっていますので、フラッシュメモリーを使用する場合には、4つのスイッチをすべてONにしてご使用ください。使用しない時はすべてのスイッチをOFFにセットしておきます。

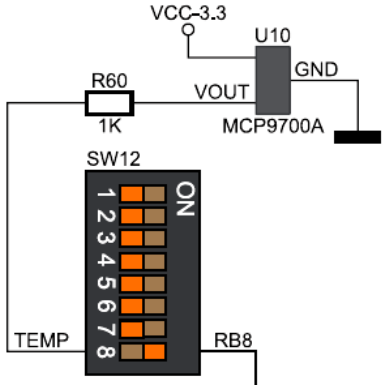


上図を参考にスイッチを設定してご使用ください。

⑤実験用温度センサ

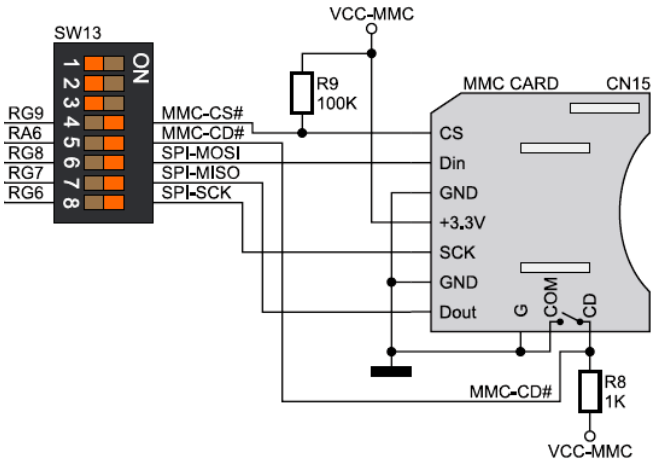
−40℃～+125℃までを計測可能なアナログリニア出力タイプの温度センサ、MCP9700Aを搭載した温度センサ回路部です。
MCP9700Aは、10mV/℃の分解能を持ち、リニアに温度に比例して電圧を出力します。(MCP9700Aの詳しい使い方についてはCD−ROMに収録されている本デバイスのデータシートをご参照ください)

PICはADコンバーターを使用してこの電圧を読み取ることで、温度データをソフトウエアで処理することができます。
MCP9700Aの出力は、SW12の8番スイッチを介して、RB8と接続されています。使用する場合には、上記スイッチをONに設定してご使用ください。



⑥MMC/SDカードスロット

MMC又はMMCと電気的な互換性のあるSDカードを、SPIモードで制御してアクセスの実験ができるカードスロットです。
メモリーカードのCSピン、CDピン、MOSIピン、MISOピン、SCKピンはそれぞれSW13を介して、PICのRG9、RA6、RG8、RG7、RG6ピンに接続されています。使用する場合には、SW13の該当部のスイッチをONにしてPICと結線し使用します。

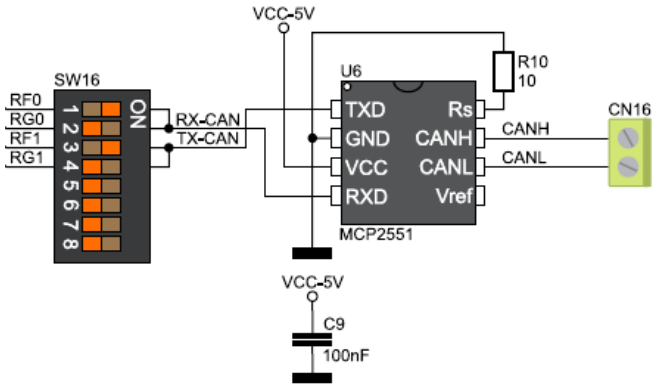


⑦リセットスイッチ

マイコンのMCLRピンと接続されており、スイッチを押すとPICにハードウエアリセットがかかります。

⑧端子台付きCANコントローラ部

CAN通信用のコントローラ、MCP2551を搭載したCAN通信実験用回路部です。CN16にはCAN−HとCAN−Lの信号線が接続されています。
MCP2551のRXピン及びTXピンの接続は、SW16のスイッチ1～4で設定することができます。RXはRF0又はRG0に、TXはRF1又はRG0に接続することができSW16で設定します。



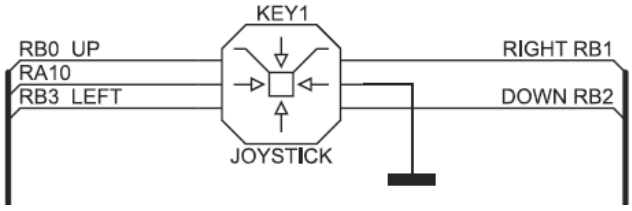
CAN通信実験をする場合には、適宜SW16を設定してご使用ください。使用しない場合には、スイッチをOFFにしておきます。

⑨320×240ドットタッチパネル付きTFT液晶ディスプレイ

解像度320×240ドットの抵抗膜方式タッチパネル付きのTFT液晶ディスプレイです。262000カラーを再現でき、きれいな画像を表示させることができます。
使用に際しては、SW16、SW17、SW18、SW19の設定が必要となります。TFT液晶のコントロールには多数の信号線がありますので、スイッチの設定も多くなります。
使用する場合には、SW17及び18はすべてのスイッチをON側に、SW19はスイッチ1番以外をON側に、4線式抵抗膜方式タッチパネルの各信号線が接続されているSW16はスイッチ5～8番をON側にセットします。詳しい回路については、CD−ROMに収録の回路図をご覧ください。

⑩ジョイスティックスイッチ

上下左右方向に動き、方向によって導通するピンが決まるジョイスティックスイッチです。
上方向の時はRB0、下方向の時はRB2、左方向の時はRB3、右方向の時はRB1、スティックを下に押した時はRA10がGNDに接地します。



使用する場合には、適宜該当ピンのプルアップの設定を行い、使用します。

⑪全I/Oピン接続ヘッダピン群

MCUカードに搭載のマイコンの全I/Oピンが取り出せるヘッダピンです。ポート単位で8ビットずつ取り出せるようになっています。

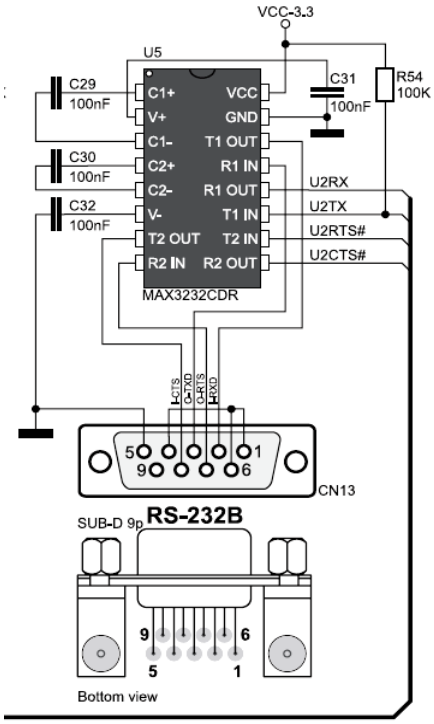
各ヘッダピンの左側には、ポート単位で8ビットずつプルアップ及びプルダウンが設定できるジャンパーJ1～J11があります。またそのジャンパー設定を各ビットに対して有効にするか、無効にするかの設定ができるディップスイッチSW1～SW11があります。

⑫2系統RS232Cポート

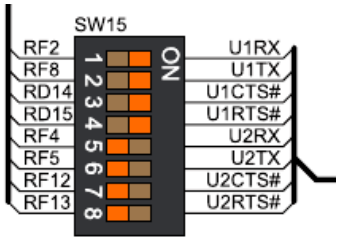
レベル変換ICを搭載したRS232Cポートです。A系統、B系統と2系統搭載しています。

A系統・B系統それぞれのTXピン、RXピン並びにフロー制御用ピンのCTSピン、RTSピンの設定は、ディップスイッチSW15によって物理的にPICのI/Oピンと接続するか、切り離すかを選択できます。スイッチをON側にセットすると、配線が接続されます。

RS232C通信を行う場合には、それぞれのスイッチを適宜設定してご使用ください。(使用しない時は必ずOFFにしておきます。)



上図はB系統側の回路図です。



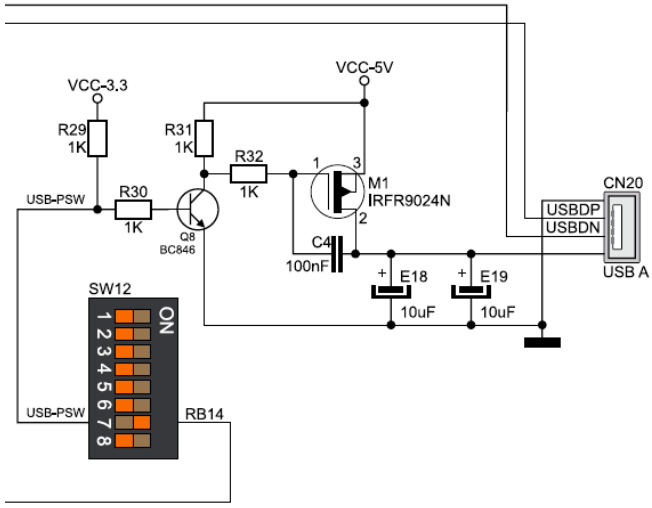
SW15の結線図です。

⑬USBホスト機能実験用USBポート

付属のMCUカードに搭載のPIC32MX460F512Lには、USBホストコントローラ機能が搭載されています。その機能を使用する場合に使うのが、本USBポート(タイプAのメス)です。USBホストとしての機能をプログラミングすることで、USB機器を制御することができます。

USBの差動信号線D+はRG2に、D−はRG3にそれぞれ接続されています。USBコネクタと直結なので、USBデバイスを装着しなければ、PIC側のピンは影響を受けませんので、ディップスイッチ等は介さずに接続されています。USBデバイスの電源となるVbasピンは電圧が+5Vであることから電源系が異なるためFET(IRFR9024N)を使用して+5Vが給電できるようになっています。

給電のスイッチとして、RB14がSW12の7番スイッチを介して接続されています。RB14によってUSBバスの電源のON/OFFを制御できます。

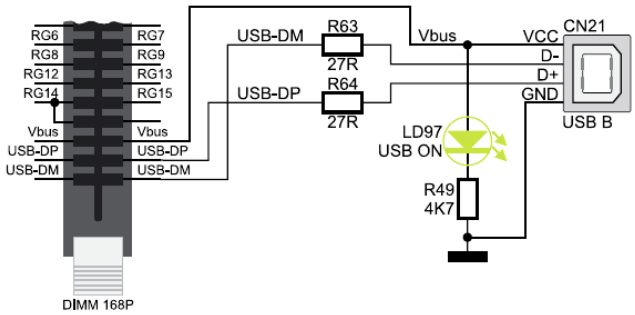


⑭実験用USBポート

PIC32MXシリーズのPICには内蔵USBシリアルエンジンが搭載されているデバイスが数多くあり、USBクライアントとして動作させることができます。

USBコネクタの差動信号線D+、D−及びVbasはDIMMスロットに接続されており、MCUカード内で、このピンをPICに接続するかどうかについて、ジャンパーチップ抵抗でセットされています。

デバイスの種類によって上記の設定が出荷時になされています。付属のMCUカードに搭載のPIC32MX460F512LはD+とD−ピン及びVUSB、VBASのピンがそれぞれPICの各ピンと接続されています。



PIC32MX460F512LはRG2及びRG3が信号線用にアサインされています。このピンは通常のI/Oピンとしては動作しませんので、ご注意ください。やむを得ずI/Oピンとして使用する場合であっても、本デバイスの仕様上、RG2とRG3は入力専用ポートになっています。詳しくはPIC32MX460F512Lのデータシートに記載されています。

⑮オーディオ信号実験用入出力回路部

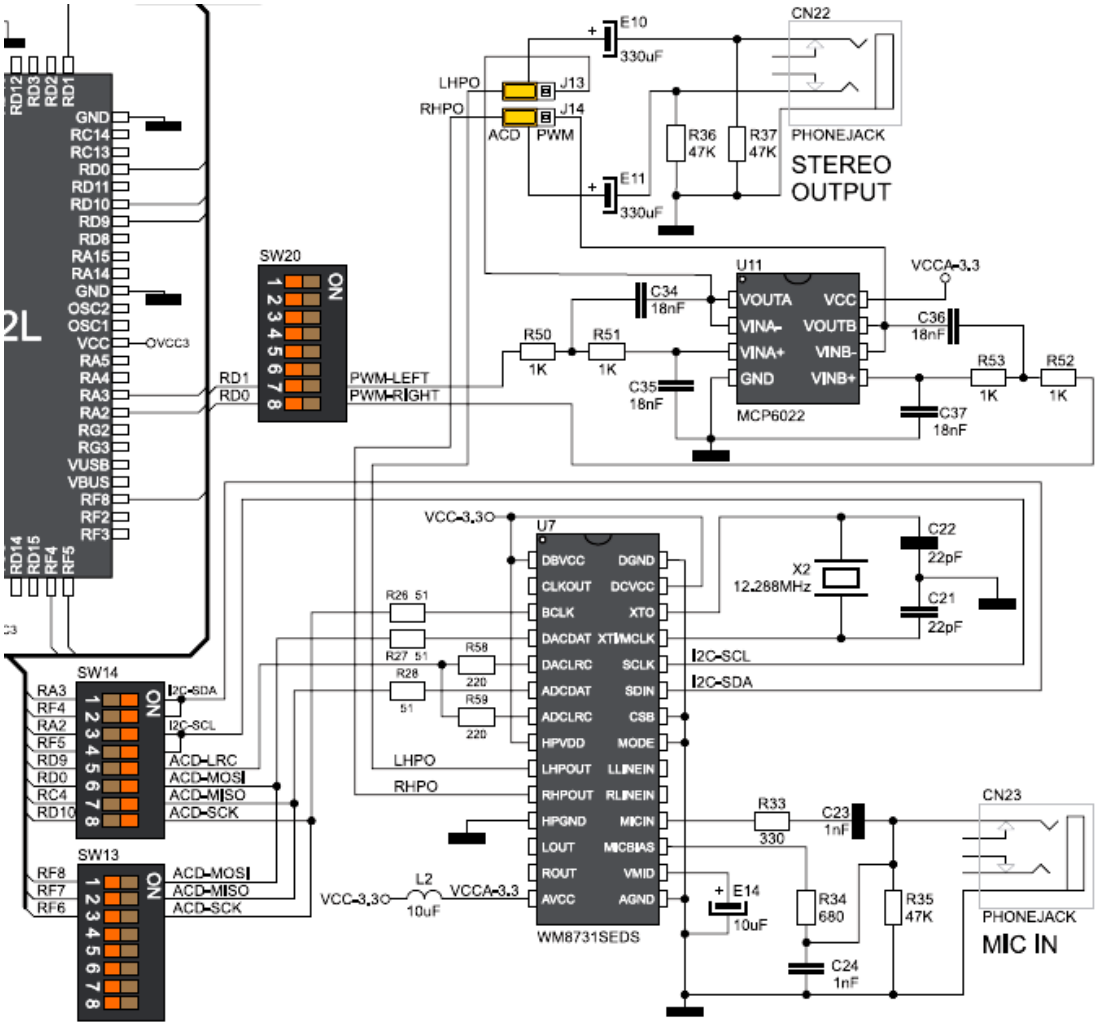
オーディオ信号のコーデックICの使い方が実験できるオーディオ信号入出力ジャック搭載のオンボード回路部です。
ヘッドフォンドライバが内蔵された高音質なオーディオコーデックICWM8731と、オペアンプMCP6022が実装されています。

WM8731は、マイク入力から入力されたアナログ信号をデジタル信号に変換(A/D)してPICへ転送するか、又はヘッドフォン出力から出力します。音量レベルは、ソフトウエアによってプログラムが可能です。マイクロフォンからの入力信号を、ヘッドフォン出力から出力する場合には、ジャンパーJ13及びJ14の設定を"ACD"側に設定します。
ジャンパーJ13とJ14は、WM8731のオーディオコーデックICから出力された音声信号をCN22のピンジャックに接続するか、又は、MCP6022のオペアンプの出力をCN22に接続するかを設定します。

また、WM8731はPICから送られるデジタルデータをDACにてアナログ信号に変換して、その音声をヘッドフォンジャックのCN22に出力させることができます。PICとWM8731のインターフェイスはSPI通信となります。
MCP6022への入力信号は、PWM信号をPICから入力します。

オーディオコーデックIC及びオペアンプを使用する場合には、ディップスイッチSW13、SW14、SW20をそれぞれ設定します。SW13の1～3番スイッチと、SW14の6～8番スイッチは同じSPI信号線の結線ですので、使用するピンに合わせて排他的に設定します。
オペアンプのPWM信号は、RD0とRD1にアサインされており、SW20の7番、8番スイッチで設定できます。

回路構成については下記をご参照ください。



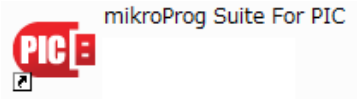
⑯16文字×2行オンボードLCD

16文字2行のオンボードLCD表示器です。LCDは+5V電源のため、隣接している74LVCC3245にて、電圧変換を行っています。
LCDはデータ線として上位4ビットを使用する4ビットモードで動作させます。LCDのデータ線D7～D4はそれぞれRB7～RB4にディップスイッチSW20を介して接続されています。
また、EnableピンはRB3、RSピンはRB2にそれぞれSW20を介して接続されています。本LCDを使用する場合には、ディップスイッチSW20の1番～6番スイッチをONにしてご使用ください。
LCDのコントラストはLCD右下に配置されているボリュームで調整できます。

書き込みソフトウエアの使用方法

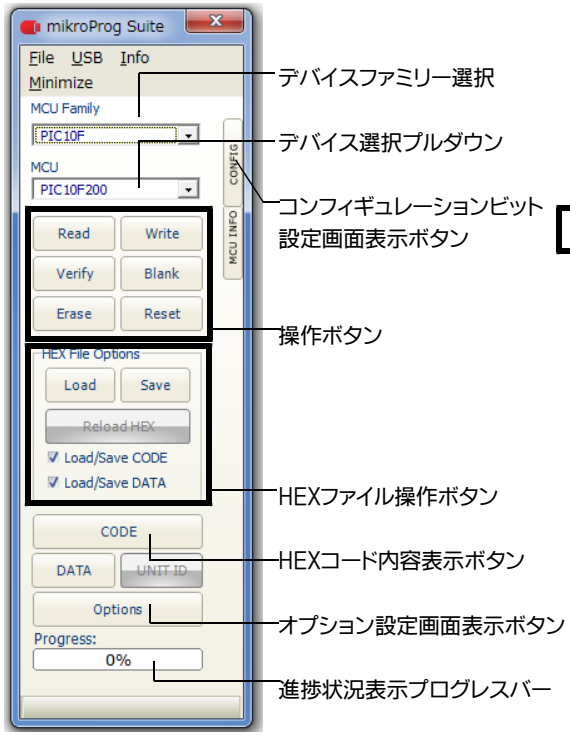
付属のUSB2.0マイコンライターの使い方を紹介します。
マイコンライターは、MPLAB等で作成したHEXファイルをPICマイコンへ書き込む際に使用するソフトウエアです。
ソフトウエアは、mikroProg Suite for PICを使用します。

■PICプログラマーの起動
PICプログラマーは、Windowsのスタートメニュー又はデスクトップに作成されたショートカットから起動できます。



スタートメニューから起動

■アプケーション概要



初期画面

"CONFIG"ボタンを押した時に右側に表示される画面

■操作の順番
次の順番でHEXファイルを書き込みます。

1 デバイス設定プルダウンより、書き込むデバイスを選択します。
最初に"デバイスファミリー選択"プルダウンからデバイスファミリーを選択します。"PIC32MX"が選択できます。

2 続いて"デバイス選択プルダウン"より書き込みを行うデバイスを選びます。

3 書き込むHEXファイルを読み込みます。
HEXファイル操作ボタンの"Load"ボタンを押します。
ダイアログが表示されますので、ファイルを選択します。

→ファイルが読み込まれるとステータスバーに、ファイル名が表示されます。

※同じファイルを再度ロードする場合には、いちいち上記のようにファイルを指定しなくても、"Reload HEX"ボタンを押すことで再読込ができます。

※ファイル名や、ファイルの配置してあるディレクトリ名に日本語や2バイト文字が含まれている場合、正しくHEXファイルの読み込みができません。ファイル名及びディレクトリ名は必ず半角英数字になるようご注意ください。

4 一般的にはこれで書き込み準備は完了ですが、必要に応じてコンフィギュレーションレジスタの設定を確認します。
コンフィギュレーションレジスタは、プログラムを書き込む時にだけしか設定できないPICマイコン全体の動作や基本的な設定などを行う特殊なレジスタです。
コンフィギュレーションビットの設定が間違っているとプログラムや回路が正しくてもプログラムは動作しません。通常、コンフィギュレーションビットの設定内容は、プログラム内に埋め込まれているため、HEXファイルを読み込んだ時点で自動的に設定されます。
しかし、設定が正しくないとプログラムの動作に問題が発生することがありますので、書き込み前に確認されることをお奨めします。

"コンフィギュレーションビット設定画面表示ボタン"をクリックします。右側に設定ウインドウが展開します。

5 必要に応じてコード保護機能も設定します。
コードプロテクションを有効にすると、一度書き込んだHEXデータは読み出せなくなります。コードを不正にコピーされることを防止できます。通常は、"None"及び"Write protection Off"に設定します。

6 設定が完了、確認したら書き込みを行います。
PICD-32MX2本体の"USB LINK"の黄LEDが点灯していることを確認してから操作ボタンの"Write"ボタンをクリックします。
書き込みを開始します。
書き込み動作中は、進捗表示プログレスバーに進捗状況が表示されます。完了するまで待ちます。

書き込みが完了すると、プログラムがすぐにボード上で動作を開始します。

■その他の機能

・Read機能
→PICマイコンからデータを読み込みます。
読み込んだデータは、"Code"ボタンで閲覧できます。またメニューバーの"File"→"Save HEX"で保存できます。

・Verify機能
→現在読み込まれているHEXファイルの内容と、PICに書き込まれているプログラムの内容が一致しているか検証します。

・Blankチェック機能
→現在装着されているデバイスのプログラムメモリがブランク(空)かチェックします。

・Erase機能
→現在装着されているデバイスのプログラムメモリの内容を消去します。

主な仕様

電源電圧:	USBバスパワー給電時 DC5V
	ACアダプタ給電時 DC9V
給電方法:	USBバスパワー又はACアダプタ
USB規格:	Ver,2.0対応
対応OS:	Windows98SE/ME/2000/XP(SP3)
	VISTA/7
対応デバイス:	PIC32MXシリーズ
	※MCUカードが対応しているデバイス
生産国:	セルビア/日本

サポート情報

PICD-32MX2のサポートを行っております。以下のいずれかの方法でご質問をお寄せください。

■FAX番号　03-3700-3548
■電子メール　support@microtechnica.net

なお、MPLAB及びMPLAB C32についてのご質問は、当方では受け付けておりません。ご質問等は、マイクロチップ社にお問い合わせください。他社製品に関することや自作回路に関するご質問にはお答え致しかねますのであらかじめご了承ください。

ソフトウエアはアップグレードされることがあります。アップグレードされると対応デバイスが増えたりバグが修正されたりします。
アップグレードの情報などは当方のwebページにてお知らせ致しますので、下記webページを定期的にご確認ください。
http://www.microtechnica.net/manual/

PICマイコン用の統合開発環境,MPLAB及びMPLAB C32はマイクロチップ社製のソフトウエアです。バージョンアップ等の情報についてはマイクロチップ社のホームページより告知されますので,下記ページを定期的にご確認ください。
http://www.microchip.com/ (英語サイト)

マイクロテクニカ
〒158-0094 東京都世田谷区玉川1-3-10
TEL: 03-3700-3535 FAX: 03-3700-3548
(C)2010 Microtechnica All rights reserved

PIC32MX2マイコンの概要

PIC32MX2シリーズは、米マイクロチップテクノロジー社が開発しているPICマイコンシリーズの1つで、CPUコアにMIPS社の32ビットコアを用いた新しいマイコンです。コアに他社製のものを採用することで、開発期間を短縮しました。

■PIC32MX2シリーズの特徴

PIC32MX2では、32ビットマイコンとしての特徴をいかせるように様々な工夫がされています。プログラムメモリーも大容量化され、より大きなプログラムが実装できます。周辺機能も豊富に搭載されています。

動作速度の高速化も特徴の1つです。動作クロックは最大80MHzとなっており、1サイクルが1クロックで動作し最大80MIPS(Million Instructions Per Second＝1秒間に何百万個の命令が実行できるかという単位)で動作します。 ただし、CPU本体がこれだけ高速に動作しても、フラッシュメモリーの動作速度はもっと遅いため、フラッシュメモリーからデータを読み出しながら実行するのではこの高速性能を生かし切れません。そこでPIC32MX2では、フラッシュメモリーとCPUの命令バスの途中にキャッシュメモリーを追加し、キャッシュメモリーからデータを取り出すことで80MHzの高速動作に追従できるよう設計されています。(ただし、キャッシュメモリーの動作はデフォルト設定では無効になっています。)

その他、割り込み処理はベクター方式となっており高速で動作させることができたり、演算機能が高速で動作したりと、従来のPICマイコンと比べ各種動作速度が格段に高速化されているのが特徴です。また、多くの周辺機能は8ビットや16ビットのPICマイコンと共通ですので、これまでにPICマイコンを使ったことがある技術者であれば、簡単に使うことができます。

■PIC32MX460F512Lの概要

PICD-32MX2に標準で付属しているPIC32MX460F512Lの概要を紹介します。このデバイスはピン数100ピンのデバイスで、プログラムメモリーは512KBと大容量です。その他ブートフラッシュメモリーを12KB搭載しています。

データメモリーは32KBです。周辺機能は、EUART/SPI/I2Cをそれぞれ2系統、分解能10ビットのADコンバータが16チャンネル、コンパレーターなどを搭載しています。

クロックはMCUカードに8MHzの水晶発振子がついており、これを利用することもできますし、内部に8MHzと32KHzの内蔵発振子を持っており、これを使うこともできます。

PIC32MXデバイスの電源電圧は、2.3V～3.6Vです。マイコン内部では、内蔵のレギュレターによって1.8Vに降圧されて電源が供給されています。

その他の詳細な機能についてはマイコンのデータシートをご覧ください。(データシートはCD-ROMに収録されています。)

MPLAB C32について

C言語は、ANSIという団体によって規格化された高級言語です。高級言語とはアセンブラ言語のように機械語に近いレベルの開発言語ではなく、より人間がわかりやすい言語体系の開発言語を指します。C言語を導入することでより簡単に、合理的にPICマイコンの開発ができるようになります。

C言語では様々なデータ型を扱えるほか、データを指し示すポインタや配列、関数などが使用でき、より高度なプログラムを記述することができます。

本PICD-32MX2では、マイクロチップ社からリリースされているMPLABという統合開発環境に、MPLAB C32というCコンパイラーを組み合わせて使用します。MPLAB C32は、製品版としても販売されていますが、"Evaluation Edition"というバージョンは無償で使用することができます。60日間は製品版と同等ですが、これを過ぎると最適化の機能が固定されます。しかし、よほど大きなプログラムを作らない限りはこの"Evaluation Edition"で十分です。

MPLAB C32は、マイクロチップ社のWEBサイトにユーザー登録すると、ダウンロードすることができます。チュートリアルを行う前にMPLABとMPLAC C32をインストールしておいてください。(本書の4ページ～5ページでその方法を解説しています。)

MPLAB C32、一般的なC言語としての機能の他にPICマイコンの機能をフルに使用できるよう様々な組み込み関数を搭載しています。組み込み関数を使用することで、様々な機能を実現できます。チュートリアルでは、そのうち一部の関数を使用して実際にプログラムを組みますが、すべてを紹介することは到底出来ません。組み込み関数についての詳細はMPLAB C32本体を解凍すると展開されるドキュメント、"32-bit-Peripheral-Library-Guide.pdf"に詳しく解説が記載されていますので、そちらをご覧ください。

チュートリアル①　～RB0の制御とボタン処理、変数の使い方～

では早速プログラムを作って開発を体験してみましょう。

■プログラムの概要

16ビットの変数を用意して、そこに入れた値を2進数でPORTBに出力するプログラムを作ってみましょう。変数を2つ用意して、外部のスイッチ入力によって、どちらの変数の値をRBに出力するのか指定できるようにしてみます。

動作クロックは高速である必要がないので、8MHzで動作させることにします。

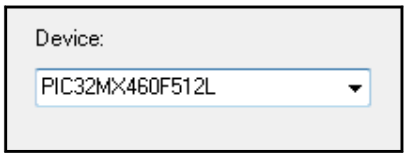
■プログラムを作成しましょう

1　MPLABを起動します。

2　MPLABでは、プログラムをプロジェクト単位で管理します。プロジェクトには、拡張子が.cのソースファイルを始めヘッダーファイルなど様々なファイルは1つのプロジェクトのもと管理する仕組みです。新しいプログラムを作る場合には必ず新しいプロジェクトを作成することをお奨めします。

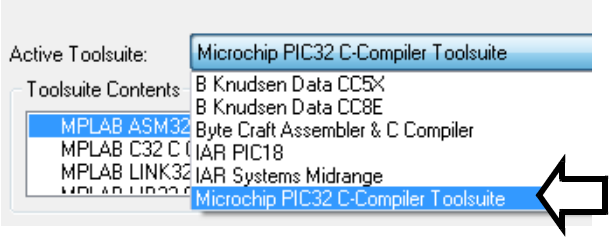
最初にプロジェクトウィザードを使用して新規にプロジェクトを作成しましょう。MPLABのメニューバーから"Project"→"Project Wizard"をクリックします。

3　ウィザードが起動しますので、"次へ"を押して続行します。
次の画面でデバイスを選びます。
プルダウンから"PIC32MX460F512L"を選びます。



"次へ"をクリックして続行します。

4　コンパイラーの種類を設定します。"Active toolsuite"のプルダウンから"Microchip PIC32 C-Compiler Toolsuite"を選択します。



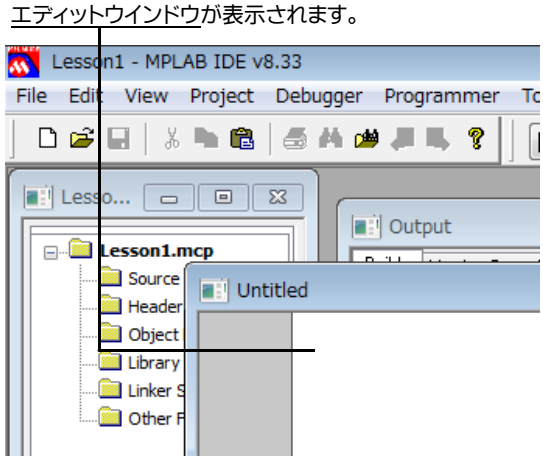
選択後、プルダウンの下の"Toolsuite Contents"に実行ファイルの一覧が表示されます。この中に×印がすべて付いていないことを確認してください。もし×印が付いている場合には、正しくMPLAB C32がインストールされていない可能性があります。MPLAB C32が正しくインストールされているか確認してください。
もし手動でディレクトリを設定する場合には、"Browse"ボタンをクリックして実行ファイルのパスを設定してください。

"次へ"をクリックして続行します。

4　プロジェクトを作成するディレクトリを指定します。
"Create New Project File"の"Browse"ボタンをクリックしてディレクトリを指定します。

5　最後の"Setp Four"はそのまま"次へ"をクリックします。
"Summary"が表示されますので内容を確認して、"完了"をクリックしてウィザードを終了します。

6　続いてソースプログラムを作成します。
メニューバーの"File"→"New"をクリックします。
エディットウインドウが表示されます。



今はまだソースプログラムに名前が付いていませんのでタイトルは"Untitled"と表示されます。

7　エディットウインドウに下記のようにプログラムを記述します。

```
#include <p32xxxx.h>
#include <plib.h>

#pragma config FPLLMUL = MUL_16, FPLLIDIV = DIV_2
#pragma config FPLLODIV = DIV_1, FWDTEN = OFF
#pragma config POSCMOD = HS, FNOSC = PRIPLL
#pragma config FPBDIV = DIV_1

void main()
{
    unsigned int a , b ;

    AD1PCFG = 0xFFFF;
    TRISB = 0;
    TRISE = 0x3;
    PORTB = 0;
    PORTE = 0;
    a = 5200;
    b = 30000;

    while(1){
        if(PORTE==1){
            LATB = a;
        }
        if(PORTE==2){
            LATB = b;
        }
    }
}
```

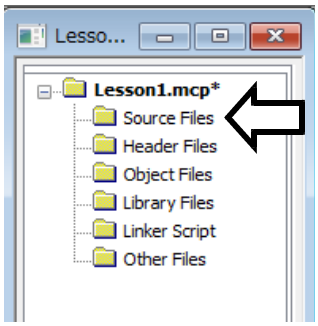
記述したらミスがないかを確認してください。
特に;（セミコロン）の抜けがないかお確かめください。

■プログラムをビルドしましょう

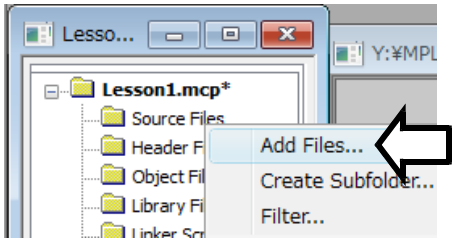
プログラムをビルドすると、hexファイルが生成されます。このhexファイルをPICマイコンに書き込むとプログラムが動作します。ビルドをするためには、プロジェクトにビルドをするファイル類をすべて追加しておかなくてはなりません。下記の手順したがってビルドをしてみましょう。

1 最初に先ほど作成したソースプログラムに名前をつけて保存します。メニューバーの"File"→"Save As"をクリックします。ダイアログが表示されますので、ファイル名を付けて保存します。なおこのときの拡張子は必ず .c としてください。
ここでは例として、lesson1.cという名前とします。

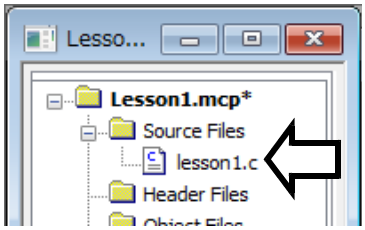
2 続いてプロジェクトに1で保存したソースプログラムを追加します。MPLABに表示されているプロジェクトツリーの一番上にある"Source Files"を右クリックします。



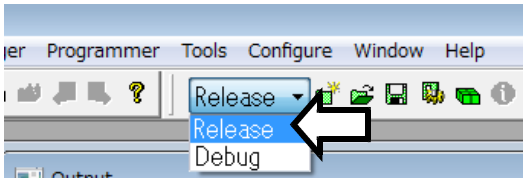
3 メニューが表示されますので、"Add Files"をクリックします。



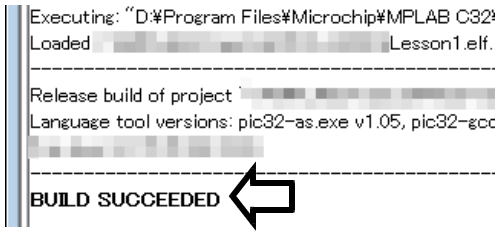
4 ファイルを開くダイアログが表示されますので、ここで先ほど手順1で保存した拡張子.cのファイルを選択して開きます。
ツリーにソースプログラムが追加されます。



5 これでビルドする準備ができました。ビルドするタイプを設定します。ツールバーにあるプルダウンから"Release"を選択します。



6 ビルドを行います。メニューバーの"Project"→"Build All"をクリックします。ビルドが開始されます。"Output"ウインドウが開き、内容が表示されます。画面一番下に"BUILD SUCCEEDED"と表示されればビルド成功です。



"BUILD FAILED"と表示された場合には、ビルドに失敗しています。エラーの原因が表示されますので修正します。
なお、"warning"は表示されていても問題ありません。"error"と表示された項目を修正してください。

hexファイルは、プロジェクトを作成したディレクトリの下に生成されています。

■PICD-32MX2の設定をしましょう

プログラムを書き込む前に、PICD-32MX2の設定を行う必要があります。PICD-32MX2では設定が正しくないとプログラムが期待したとおりに動作しなかったり予期せぬ動作をしてしまいます。プログラム書き込みの前に必ずプログラムの内容とボードの物理的な設定が正しいかどうかを確認する必要があります。

1 このプログラムではPORTBのLEDとPORTEのタクトスイッチしか使用しません。よって、その他の周辺回路とはすべて切り離しておく必要があります。
最初にボードの全部のディップスイッチのスイッチをOFF側に設定してください。PICD-32MX2には全部で20個のディップスイッチがあります。それらすべてをOFFにします。

2 続いて、PORTBのLEDをONにします。
SW12の"PORTB"と書かれたスイッチ(2番)だけONにセットします。

3 PORTEのI/Oピンは定常時Lowで、タクトスイッチが押された時にHighになるようにします。
タクトスイッチ群の左下にあるJ15のジャンパーを"VCC3"側にセットします。
ジャンパーJ8を"Pull-Down"側にセットし、右隣にあるSW8のディップスイッチ1番と2番のみをONにセットします。
これで、RE0とRE1はアクティブHの設定となりました。

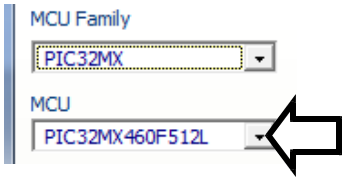
設定はこれで完了です。プログラムを書き込む前には必ずボードの状態を正しく設定しておく必要があります。設定が間違っていると、プログラムの内容が正しくても、動作は正常でなくなったり、予期せぬ方向に電流が流れたりして、思いもよらない動作をすることがあります。

■プログラムを書き込み、動作を確認しましょう

ボードの設定が完了したら、生成したhexファイルを書き込みソフトウエアでPICマイコンに書き込みます。書き込んだ直後からプログラムが動作します。下記の手順で書き込みます。

1 Windowsのスタートメニューから"スタート"→"プログラム"→"Mikroelektronika"→"mikroProg Suite For PIC"→"mikroProg Suite For PIC"の順でクリックして起動します。

2 最初にデバイスの種類を設定します。
デバイスプルダウンからPIC32MX460F512Lを選択してください。

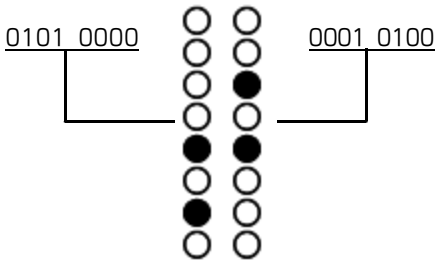


3 先ほど生成したhexファイルを読み込みます。
"Load"ボタンをクリックします。ファイルを開くダイアログが表示されますので、先ほど作ったhexファイルを指定して開きます。
hexファイルは、プロジェクトを作成したディレクトリと同じディレクトリに作成されています。

4 一般的にhexファイルを書き込む前にはコンフィギュレーションビットの値を設定しますが、先ほどの例では重要な設定項目はソースプログラム内に記述していますので、hexファイルを読み込んだときに自動的にコンフィギュレーションビットの値は設定されます。

5 プログラムを書き込みますので、"Write"ボタンを押します。
書き込みが開始されます。(書き込みは数秒で終わります)

6 書き込みが完了したら、動作確認をしてみましょう。
まず、RE0のタクトスイッチを押します。RE0の時は、変数aの値、すなわち5200がPORTBに代入されますので、5200を2進数に変換すると、 0001 0100 0101 0000 です。LEDを見てみましょう。
下図のように点灯しているはずです。



値の通りLEDが点灯していることがわかります。
同様にしてRE1のスイッチも押してみましょう。RE1を押すと変数bの値(30000)が代入されますので、0111 0101 0011 0000 という値どおりLEDが点灯するはずです。

このほかに値を変えて試してみましょう。
なお最大値は16ビットの最大値 1111 1111 1111 1111 ですので、PORTBに0xFFFFを代入すると、LEDが全点灯となります。

■プログラムの解説

#include<p32xxxx.h>
#include<plib.h>
拡張子.hは、ヘッダーファイルといい各種定義などがひとまとまりになっているファイルです。PIC32MX2を使う場合には共通のヘッダーファイルとしてp32xxxx.hというファイルがあります。このヘッダーファイルをインクルードしておくと各種レジスタや、ピット操作にレジスタの名称を使うことができます。
plib.hファイルは、周辺機能を使うためのライブラリです。
インクルードはプログラムの先頭で行うようにします。

#pragma config
コンフィギュレーションビットは、クロックの種類を設定したりクロック周波数の分周比を設定したり、ウォッチドッグタイマーの設定をしたりするPICマイコンの基本動作について設定を行うものです。
通常コンフィギュレーションビットは、hexファイルを書き込む時に書き込みソフトウエアで設定して書き込むものですが、いちいち書き込む際に設定をするのでは面倒ですし、間違ってしまうこともあります。そこでソースプログラムにあらかじめコンフィギュレーションビットの設定値を記述しておくことで、hexファイルを書き込みソフトウエアが読み込んだときに自動的にコンフィギュレーションビットの値を設定させるプラグマが用意されています。
※pragmaはプリプロセッサキーワードです。

"#pragma config"に続けてコンフィギュレーションビットのパラメーターと設定値を記述します。記述の仕方、項目の一覧はMPLAB C32のインストールされたディレクトリの下にあるDOCディレクトリに展開されるヘルプファイル "hlpPIC32MX2ConfigSet.chm"に型式ごとに記述がありますので、こちらを参考にしてください。

void main(){…}
main関数です。C言語では1つのプログラム内に必ず1つのmain関数が必要です。voidはmain関数の型で戻り値がない関数であることを宣言しています。{ }で囲まれた範囲がmain関数の範囲となります。

unsigned int a , b ;
符号なしのint型変数を宣言しています。C言語では文の最後には必ず ; (セミコロン)をつけます。
符号なしのint型の場合扱える値は、0～65535までです。

AD1PCFG = 0xFFFF;
AD1PCFGレジスタは、ADコンバータ用のアナログ電圧入力ピンを設定するレジスタです。該当のビットが0でアナログ入力、1でデジタルI/Oに設定されます。アナログ入力ピンはPORTBにアサインされていますので、ここでは、PORTBすべてのビットがデジタルI/Oになるよう0xFFFFを代入しています。

TRISB = 0 ;
TRISE = 0x3;
TRISレジスタは、ピンの入出力方向を設定するレジスタです。
該当のビットが0の場合は出力、1の場合は入力となります。本チュートリアルではRBは全ピン出力、REはRE0とRE1が入力なので0x3として設定しています。
C言語では16進数の値を表現する時は数値の前に0xを付けます。

PORTB = 0 ;
PORTBに0を代入しています。 =(イコール1つ)は代入演算子で右辺の値を左辺に代入します。mikroCでは「レジスタ名=値」の書式でレジスタに値を代入できます。

a = 5200 ;
b = 30000 ;
int型変数に値を代入しています。

while(1){…}
while文は( )内の値が1又は真であれば{ }内の処理を繰り返す繰り返し文です。ここでは評価式を1としていますので永久に{ }内が実行されます。
C言語では処理を永久ループさせたい場合にはよくwhile文を使用します。

if (PORTE==1){ ;

if文を使用した入力判定部分です。

()内の評価式が真の時にその後ろに記述された文が実行されます。==(イコール2つ)は比較演算子の「等しい時」を表す演算子です。よくある間違いとして=を1つしか記述しない場合があります。=が1つの場合には代入演算子となってしまい正しく動作しません。文法上は間違いではないのでエラーがでないため間違いを見逃してしまいプログラムが思うように動かない・・・ということがあります。

ここでは、「PORTBが1になった時」という条件判定をしています。PORTBが1になるとき・・というのはRB1のスイッチが押された時です。

LATB = a;

LATBレジスタに変数aの値を代入しています。

LATレジスタとは、出力ポートの制御に使うためのレジスタで、出力として使用する場合には、PORTレジスタと同じ動作をします。よって、この文は下記のように書き換えることもできます。

PORTB = a;

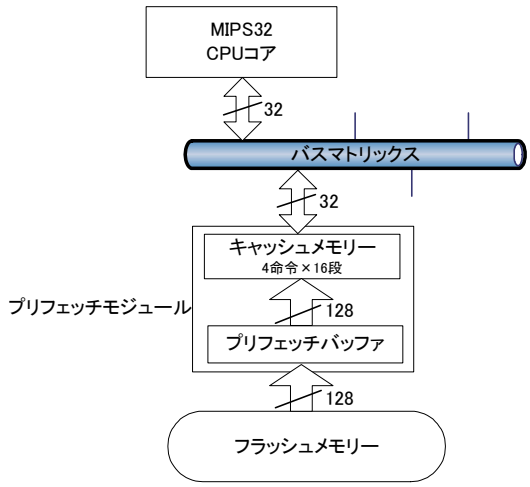
LATレジスタは出力として使用するときにはあまり意識しなくてもよいのですが、入力として使用するときには、注意が必要です。PORTレジスタは入力ピンの状態をそのまま読み込みますが、LATレジスタは、入力の時にピンの物理的な状態を読み込みません。LATレジスタはポートのLatchの状態を読み込むだけです。

ここまででチュートリアル①は終わりです。一通りハードウエアの使い方とソフトウエアでの開発を体験しました。ここまでの一連の操作は今後すべてのプログラム開発で共通ですので、しっかりと手順と方法を覚えておくことが重要です。これ以降のチュートリアルでは、プログラム本文とPICD-32MX2ボードの設定、プログラムの解説は掲載していますがソフトウエアの操作については解説していませんので、わからなくなった時はチュートリアル①を読み直してください。

チュートリアル②　～PIC32MX2のスピードを体験する～

PIC32MX2は、これまでの8ビットや16ビットのPICと異なり、1サイクルが1クロックとなっており、クロック周波数は最大80MHzまで出せる高速MPUとなりました。これは、コンピューターの性能指標の1つとして用いられるMillion Instructions Per Second(MIPS)という単位でいうところの80MIPSと表記できます。パソコン初期によく使われたインテル社の486DXというCPUは、54MIPSですので速度だけで見ればPIC32MX2は初期のパソコンをしのぐ早さといえます。

しかし、1つ問題があります。せっかくCPUの速度は80MHzと高速なのにプログラムを記憶するフラッシュメモリーの速度は最高30MHzなので、その高速性能に追従できないのです。そこでPIC32MXシリーズでは、CPUとフラッシュメモリーの間にキャッシュメモリーを挟むことでその問題を解決しています。キャッシュメモリーは高速動作が可能なので、CPUの速度に間に合うのです。キャッシュメモリーは、4つの命令を16段格納することができ、最大64個の命令を格納できます。よって、あらかじめキャッシュメモリーに命令が入っていれば(プリフェッチといいます)CPUは80MHzという高速動作でも次々とキャッシュメモリーからデータを取り出してノーウェイトで実行できるのです。概念図を示します。



しかし、このプリフェッチモジュールのキャッシュメモリー機能は、電源投入時のデフォルト状態では、無効になっておりその効果を体験できません。ここでは、このキャッシュメモリー機能の有無によってどれだけ動作速度に差があるのかを実験してみることにします。

本チュートリアルで、外部8MHzの発振子から80MHzを生成する内蔵PLL機能の設定方法の他、プログラム内でディレイ(遅延)を作る方法を学習します。本チュートリアルで用いる方法で、プログラム内で用いる時間遅延を作ることができます。なお、本チュートリアルの実験にはオシロスコープをご用意ください。

■プログラムの作成

```
#include <p32xxxx.h>
#include <plib.h>

void main()
{
    SYSTEMConfigPerformance(80000000) ;

    AD1PCFG = 0xFFFF;
    TRISB = 0;

    while(1){
        LATB = 0x00;
        LATB = 0x01;
    }
}
```

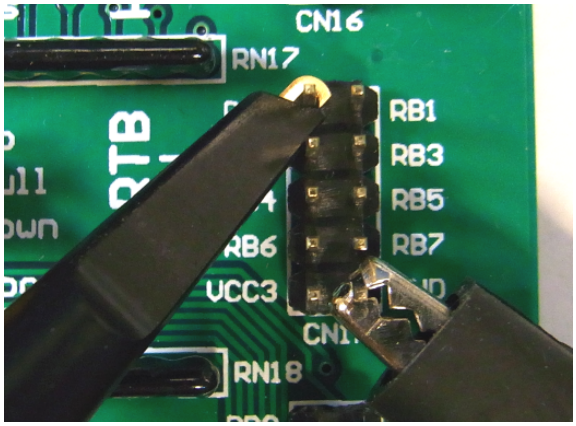
プログラムを記述したら、"Build All"でhexファイルを作ります。書き込みの前にボードの設定をしましょう。今回は手動でコンフィギュレーションビットの値を設定します。

■PICD-32MX2の設定をしましょう

PICD-32MX2ボード上のすべてのディップスイッチはOFFに設定してください。

■オシロスコープを接続しましょう

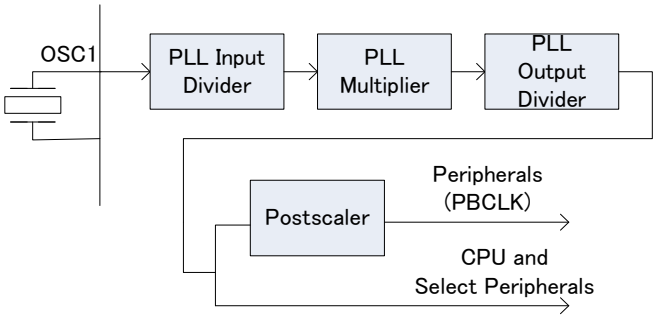
オシロスコープをお持ちでしたらRB0にプローブを取り付けましょう。PICD-32MX2ボードの右側にはすべてのI/Oと接続できるヘッダーピンがあります。RB0のピンにプローブを取り付けます。



RB0のパルスを観察します。電圧の振幅は3.3Vp-pですので、オシロスコープのレンジを調節してください。時間軸は20ns/div程度がよいでしょう。

■コンフィギュレーションビットを設定しプログラムを書き込みます

今回は、コンフィギュレーションビットの値を設定します。設定としてはクロック周波数を80MHzに内蔵PLLで逓倍することです。PIC32MXのオシレーター設定は複雑です。概念図を下図に示します。



外部の8MHzの水晶発振子のクロックは最初にPLL入力ディバイダに入ります。ここでクロックは分割され、周波数が低くなります。そのクロックはPLLに入り逓倍され、PLL出力ディバイダでさらに分割されます。そのクロックがCPUに供給される他、周辺用にはさらにポストスケーラーで分周したクロックが供給されます。

8MHzのクロックから80MHzのクロックを作るには、最初のPLL入力ディバイダでクロックは1/2にし、そのクロックをPLLで20倍します(8MHz÷2×20=80MHz)。PLL出力ディバイダでは1/1として分割せずそのままCPUに80MHzのクロックを供給することにします。

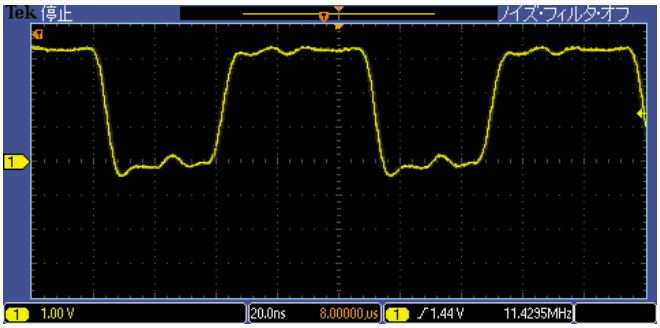
書き込みソフトを起動します。最初にhexファイルを読み込みましょう。"Load"ボタンを押して生成したhexファイルを読み込みます。続いて、"Configuration Bits"の部分を設定していきましょう。下記のように設定してください。なお記載のない項目は変更せずそのままにしておきます。

・System PLL Output Clock Divider	PLL Divice by 1
・PLL Multiplier	20x Multiplier
・PLL Input Divider	2x Divider
・Watchdog Timer Enable	WDT Disabled
・Peripheral Bus Clock Divisor	PBCLK is SYSCLK div 1
・CLKO Enable	CLKO output Disabled
・Primary Oscillator Configuration	HS
・Oscillator Selection	Primary with PLL Module

設定ができたら"Write"ボタンをクリックして書き込みます。
設定がややこしいですので間違えないように注意してください。

■波形を観察してみましょう

オシロスコープの波形を観察してみましょう。下図は、オシロスコープの波形です。H期間がL期間より若干長い矩形波が観察できます。周波数は約11.43MHzとなります。



波形はやや歪んでいますが、L期間は約37.5ns、H期間は約50.0nsとなっています。なぜこのようなパルスとなるのでしょうか。このなぞの解決をするには、Cコンパイラーが作るアセンブラ言語のファイルを見てみる必要があります。C言語で書かれたプログラムはアセンブラ命令で構成されます。このアセンブラ命令の構成を見てみましょう。

MPLABのメニューバーより"View"→"Disassembly Listing"をクリックします。スクロールバーを操作して下の方に移動すると、while文の内容を見ることができます。

```
19:
20:                     while(1)
21:                     {
22:                         LATB = 0x00;
9D0004CC  3C02BF88    lui       v0,0xbf88
9D0004D0  AC406060    sw        zero,24672(v0)
23:                         LATB = 0x01;
9D0004D4  3C03BF88    lui       v1,0xbf88
9D0004D8  24020001    addiu     v0,zero,1
9D0004DC  AC626060    sw        v0,24672(v1)
9D0004E0  1000FFFA    beq     zero,zero,0x9d0004cc
```

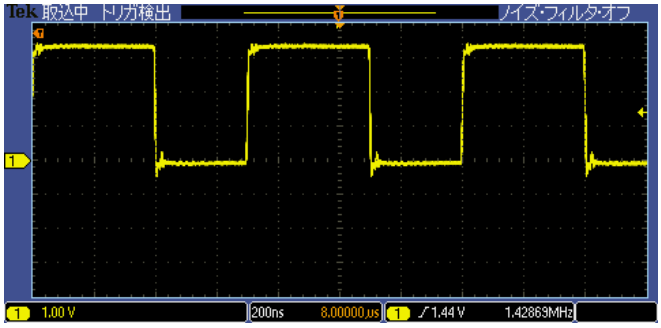
この内容を見るとLATB=0x00の後ろに2つのアセンブラ命令があります。80MHzで動作しているということは1サイクル12.5nsとなります。最初のLATB=0x00は3サイクルですので、12.5ns×3=37.5nsとなります。LATB=0x01の後ろは4つですので12.5ns×4=50nsとなり、オシロスコープの波形と一致します。全体では、7サイクルとなりますので、12.5ns×7=87.5nsとなり逆数にすると約11.43MHzとなります。

このようにC言語のプログラムを見ただけではわからない細かな動作内容もアセンブラの内容を見ることで理解することができます。ちなみにプログラム中の時間遅延はこのようにアセンブラ命令を繰り返し実行して時間稼ぎをして作り出すのが一般的です。

※オシロスコープで観察する波形が大きく歪んでいたり、グリッチがある場合には、オシロスコープのプローブをヘッダーピンコネクタではなくMCUカード下側にあるI/Oピン取り出しパターンの"RB0"の部分に当ててみてください。PICD-32MX2では、ボード内で配線が引き回されているため高速な矩形波の場合、波形が鈍ることがあります。

■キャッシュを使わない設定にして波形を観察してみましょう

今度はキャッシュメモリーを使用しない設定にして同じように波形を観察してみましょう。プログラム中の "SYSTEMConfigPerformance(80000000);" の部分の前に//(スラッシュ2つ)を入れてコメント行としてこの部分を実行しないようにします。そして再度Build Allでhexファイルを作り、先ほどと同様のコンフィギュレーションビットの値でプログラムを書き込み波形を観察してみましょう。



周波数は約1.43MHzとなりました。L期間は約300ns、H期間は400nsとなりました。先ほどの場合と比べると8倍遅くなっていることがわかります。"SYSTEMConfigPerformance();" という関数はキャッシュメモリーを有効にして80MHzの性能を出せるようにキャッシュメモリーを設定するためのものです。キャッシュメモリー使用の有無によって、8倍の時間差が生じることがわかりました。

同様にして、次はコンフィギュレーションビットの値を色々と変更して波形を観察してみてください。例えばPLLの倍率を24倍から15倍に変更してみましょう。クロックは50MHzとなりますので、動作も変わるはずです。オシロスコープで波形を観察しながら色々と変更してみてください。

チュートリアル③　～TFT液晶への文字表示とジョイスティック～

PICD-32MX2に搭載のカラーTFT液晶ディスプレイに文字列を表示し、その文字列をジョイスティックにて上下動させるプログラムを作ります。

本来TFT液晶の駆動方法は、かなり複雑で難しいですが、ライブラリを使用することで簡単に使用することができます。本プログラムでは、チュートリアル用のベースプロジェクトと各種サンプルプログラムをパソコンにインストールして使用します。ベースプロジェクトファイルを使うことで、インクルードすべきファイルが既に登録されていますので簡単にプログラムが作成できます。

■プログラム作成前の準備

プログラム作成前にPICD-32MX2用のサンプルプログラム集をパソコンにインストールします。本プログラム集はMPLAB C32で作られたPICD-32MX2用のサンプルプログラムで、各種周辺回路を利用するためのライブラリが含まれています。また、PIC32MXデバイス用にマイクロチップ社からリリースされている各種ライブラリも同時にインストールされます。(マイクロチップ社のMPLAB C32用ライブラリはマイクロチップ社のWEBサイトからもダウンロードできます。)

複雑なプログラムについては、すべてのプログラムを自分で記述するのは大変です。main関数を含むソースプログラムであらかじめ作られているライブラリをインクルードすることで、ライブラリを追加し利用すると比較的簡単にプログラムを作ることができます。

1 CD-ROMをパソコンのドライブにセットし、"C32サンプルプログラム"フォルダを開きます。

2 "Setup.exe"をダブルクリックして実行します。
→インストールが開始されます。

※インストールディレクトリはインクルードファイルがある関係上、パスを変えてしまうと正しくインクルードファイルのパスが設定できない関係上、ユーザーは指定することができません。インストールされると、Cドライブの直下に"LV32MX Examples v1.00"というフォルダが作られます。フォルダ名を変えないようにしてお使いください。

■プログラムの作成

PICD-32MX2に装着されているカラーTFT液晶に文字列を表示させます。ジョイスティックスイッチを上下動させるとそれに伴って文字列が上下動します。また左右にジョイスティックスイッチを動かすと文字列の色を変更できるようにします。

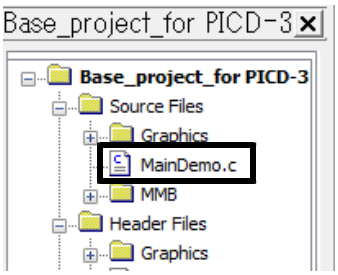
最初にベースとなるプロジェクトファイルをオープンしましょう。

1 MPLABを起動します。メニューバーの"Project"→"Open"を選択します。

2 ダイアログが表示されますので、先ほどインストールしたチュートリアルがインストールされている"LV32MX Examples v1.00"フォルダを選択して、その中にある"Base_Project"フォルダを開きます。

3 その中にある"Base_project_for PICD-32MX2.mcp"を選択して「開く」ボタンを押してプロジェクトを開きます。

4 プロジェクトが開いたら、プロジェクトに含まれているファイルのツリーウインドウから"Source Files"ツリーの下にある"MainDemo.c"をダブルクリックして開きます。



5 この"MainDemo.c"ファイルにプログラムを記述していきます。

"MainDemo.c"ファイルを見ると、"void main(){ "の部分があり、その下に"//User code here!"と書かれた部分があります。この部分にコードを記述していきます。次のようにプログラムを記述しましょう。
なお本書では、void main()関数内だけの内容を紹介しています。その他の部分はそのまま使用します。

プログラムは次のように記述しましょう。

```
void main(){
  char c,i=0;
  char txt[] = "This is test string..";

  init_picd32mx2();

  LCDCenterString(i, txt);
  MMBFadeIn(100);

  while(1){
    c = MMBGetKey();

    switch (c){
        case JOY_DOWN:
            i++;
            LCDClear();
            LCDCenterString(i, txt);
            MMBFadeIn(100);
            break;

        case JOY_UP:
            i--;
            LCDClear();
            LCDCenterString(i, txt);
            MMBFadeIn(100);
            break;
    }
  }
}
```

プログラムを記述した後は"Build All"でビルドをして、hexファイルを作ります。インクルードファイルが多いですので、ディレクトリに誤りがあると、正しくビルドできません。

ビルドを行うと、HEXファイルは下記のディレクトリに作られます。

C:¥LV32MX Examples v1.00¥Base_Project¥bin

■PICD-32MX2の設定

PICD-32MX2上ではTFT液晶ディスプレイとジョイスティックスイッチを使用します。それ以外は使用しません。下記のように設定してください。

ディップスイッチSW17、18、19はすべてのスイッチをON位置に設定します。それ以外のスイッチはすべてOFF位置に設定します。

■プログラムの動作確認

HEXファイルを書き込むと、TFT液晶の上下中心部分に"This is test string.."と白文字で表示されます。

その状態で、ジョイスティックスイッチを上に動かすと文字例が上に移動します。下に動かすと文字列が下に移動します。

■プログラムの解説

char txt[] = "This is test string..";

char型の配列に文字列を代入しています。C言語では文字はASCIIコードにて扱われます。

通常配列を宣言する際には要素数(インデックス値)を[ ]内に入力しますが、[ ]内を空欄にすると自動的に要素数が設定されます。

配列や変数を宣言する際に初期値を代入できますが、char型に限り文字列で初期化できます。"(ダブルクォーテーション)で囲むと文字列として認識されます。

init_picd32mx2();

PICD-32MX2ボードの初期設定を行う関数をコールしています。

LCDCenterString(i, txt);

LCDCenterString関数は指定した縦の位置に、文字列を表示する関数です。PICD-32MX2用に作られた関数です。

変数iには初期値0が入っています。本関数は、位置指定0の時、液晶ディスプレイの中心部に文字列を表示します。値が正の整数の場合文字列は中央部より下に表示されます。逆に値が負の場合には、中央部より上部に表示されます。この仕組みを利用して、変数iの値が変わることで、文字の表示位置を変えています。

MMBFadeIn(100);

PICD-32MX2用に作られた関数です。

TFT液晶に文字列を表示する時の時間をミリ秒で指定します。値を大きくすると、文字表示がゆっくりになります。

c = MMBGetKey();

PICD-32MX2用に作られた関数です。

ジョイスティックスイッチの位置によってchar型の文字列を返します。下に下げると"JOY_DOWN"という文字列が返ります。同様にして上下左右のスイッチ情報を変数で受けることができます。

ここでは、switch文を用いて条件分岐しています。

switch (c){

switch文です。( )内に判定式を記述して、以降のcase:に対応する条件式に処理を分岐します。ここでは変数cの値、すなわちジョイスティックスイッチの方向文字列によって処理を分岐しています。

なおcase内に記述したルーチンの最後には必ずbreak文を付けます。

チュートリアル④ ～TMR1割込を使用してLEDを点滅～

PIC32MXシリーズのデバイスには、TMR1という内蔵タイマーが搭載されています。TMR1は、16ビットのカウンターで指定したクロックに同期させてカウントを行わせることができます。

TMR1は、PR1レジスタの値と常に比較されるようになっており、TMR1の値と、PR1レジスタの値が一致した時、割込が発生する仕組みになっています。また割込発生後は、TMR1の値は0に初期化されます。

■PIC32MXシリーズの割込について

PIC32MXシリーズでは、割込の種類(何のイベントで発生した割込か)によって、ジャンプするベクタ(ジャンプベクタと呼びます)が異なるベクタ方式という割込を採用しています。8ビットのPICマイコンに比べ格段に構造が複雑になっています。

これにより、様々な割込に応じて異なる割込ルーチンを作成でき、より複雑な割込プログラムを作ることができます。

ジャンプベクタは、プログラムメモリーのアドレスに割り当てられています。割込が発生すると、その割込の種類によって、ジャンプベクタが変わります。どんな割込が発生した時、どのジャンプベクタへジャンプするのかは決められており、例えば今回使用するTMR1割込の場合には、ベクタ番号は4となります。このように割込の種類と、飛び先アドレスの一覧を主ベクタテーブル(IVT)と呼びます。割込ベクターテーブルの一覧は、PIC32MXシリーズのデータシートに記載がありますので、併せてご覧ください。

PIC32MXシリーズでは割込に順位が付きます。従来の8ビットのPICでは、割込は同時には1種類だけしか選択できませんでしたから、発生した割込の順位などは最初から必要ありませんでした。しかしPIC32MXでは複数の割込をベクタ方式で処理できるため、割込に優先順位を持たせてあります。

優先順位は、7段階で7が最も優先順位が高い割込になります。レベル7の割込の場合は、シャドーレジスタが使用されて、レジスタの退避と復旧が行われるようになっており、高速化を実現しています。

MPLAB C32には割込を使用するために利用できる関数が数多く用意されており、プログラムが作りやすくなっています。また割込発生時の処理は___ISRというマクロが用意されており、これを使うことでより簡単にプログラムの作成ができます。

■プログラムの作成

TMR1を利用してRB0に接続されたLEDを約1秒間隔で点滅させるプログラムを作成します。本プログラムは、新しいプロジェクトで新規に作成します。MPLABのメニューバー"Project"→"Project Wizard"から新規に作成してください。

```
#include <plib.h>

#pragma config FPLLMUL = MUL_20
#pragma config FPLLIDIV = DIV_2
#pragma config FPLLODIV = DIV_1, FWDTEN = OFF
#pragma config POSCMOD = HS, FNOSC = PRIPLL
#pragma config FPBDIV = DIV_8

int main(void)
{
  AD1PCFG = 0xFFFF;
  TRISB = 0;
  PORTB = 0;

  OpenTimer1(T1_ON | T1_SOURCE_INT | T1_PS_1_256, 39062);
  ConfigIntTimer1(T1_INT_ON | T1_INT_PRIOR_7);
  INTEnableSystemMultiVectoredInt();

  while(1);
}

void __ISR(_TIMER_1_VECTOR, ipl7) Timer1Handler(void)
{
      mT1ClearIntFlag();
      mPORTBToggleBits(BIT_0);
}
```

プログラムが書き終わったら、PICD-32MX2の設定を行います。今回はPORTBのLEDしか使用しませんので、ディップスイッチSW12の"PORTB"(2番スイッチ)だけをONに設定して他のスイッチはすべてOFFに設定してください。設定が完了したらビルドしてhexファイルを書き込みます。

RB0のLEDが1秒間隔で点滅すれば成功です。

■プログラムの解説

このプログラムではTMR1の割込を使用して1秒間隔の点滅を作っています。TMR1は16ビットのカウンターです。本プログラムでは、TMR1のカウントアップのためのクロックには、ペリフェラルバスクロック(PBCLK)を使用します。

システムクロックは80MHz(外部8MHzクロック÷2×20=80MHz)ですので、それをPBCLKディバイダによって1/8にして8MHzとします。これをさらにTMR1のプリスケーラによって1/256にします。すると周波数は　80MHz÷8÷256 = 39.0625KHzとなります。すなわち25.6μ秒ごとにカウントアップされていくことになります。1秒を作りたいので、PR1レジスタに39062を代入しておけば、TMR1のカウンタが39062回目で割込が発生します。これはすなわち、25.6μ秒×39062=999.9872m秒ということで約1秒毎の割込となります。

TMR1は割込発生後は0にクリアーされます。

OpenTimer1(T1_ON | T1_SOURCE_INT | T1_PS_1_256, 39062);

OpenTimer1()関数はTMR1の動作モードを設定する関数です。
引数"T1_ON"でTMR1が有効になります。クロックのソースは、内蔵発振子を使用しますので、"T1_SOURCE_INT"を指定します。
プリスケーラは、分周比1:256にしますので、"T1_PS_256"にします。最後にPR1レジスタに設定する上限値を記述します。ここでは39062を代入しています。

ConfigIntTimer1(T1_INT_ON | T1_INT_PRIOR_7);

ConfigIntTimer1()関数は、TMR1の割込条件を設定します。
"T1_INT_ON"で割込を許可します。
続いて、割り込みレベルを設定します。値は0～7です。7が最も割込順位が高くなります。

INTEnableSystemMultiVectoredInt();

ベクタ方式の割込を有効にしています。

void ___ISR(_TIMER_1_VECTOR, ipl7) Timer1Handler(void)

___ISRは割り込み処理ハンドラ作成用のマクロです。
書式は下記の通りです。

```
void ___ISR(vector, iplx)  XXXHandler(void){
          ハンドラの処理
          mXXXClearIntFlag();
}
```

このうち、vectorで指定する部分はベクター番号か、あらかじめ決まっている文字列を入力します。TMR1の場合には上記のような文字列の記述でもいいですし、ベクター番号4を入力してもよいことになります。
XXXNameのXXX部分も文字列が切れられており、TMR1の場合には"Timer1"としなくてはなりません。
割込の順位はiplxのxの部分に値を記述します。ここでは最も順位の高い7を記述しました。

この記述にすることで、TMR1の割込が発生した時はこの関数にジャンプするということが明示的に記述できるわけです。

mT1ClearIntFlag();

割り込みルーチンでは必ず記述します。TMR1割込フラグをクリアーします。このフラグをクリアーしないと割込がずっと発生していることになってしまい、新たに割込が発生しなくなります。
割込発生フラグはソフトウエアで必ずクリアーします。

mPORTBToggleBits(BIT_0);

I/Oピンに関する関数です。

指定したビットの状態をトグル(切り替え)します。HならばLに、LならばHに切り替えます。これを割込発生毎に実行するため、RB0が点滅します。

チュートリアル⑤　～非同期式シリアル通信(UART)～

PIC32MX2シリーズでは非同期式シリアル通信(UART)モジュールを2チャンネル搭載しており、ハードウエアによるシリアル通信が可能です。本チュートリアルでは、UARTにてデータを送受信するプログラムを作ります。なお、シリアル通信の通信速度は115.2kbps、データ長8ビット、パリティなし、1ストップビットという標準的なプロトコルとします。

※本プログラムの動作確認のためには、パソコンのRS232CポートとPICD-32MX2のRS232Cポートを全結線ストレートケーブルで接続する必要があります。

■プログラムの作成

UART1を利用し、文字"a"以外のデータを受信したら、そのまま受信したデータを送り返すプログラムを作ります(エコーバックといいます)。また文字"a"を受信した時は、"HELLO"という文字列を送信するようにします。

```
#include <plib.h>

#pragma config FPLLMUL = MUL_20
#pragma config FPLLIDIV = DIV_2
#pragma config FPLLODIV = DIV_1, FWDTEN = OFF
#pragma config POSCMOD = HS, FNOSC = PRIPLL
#pragma config FPBDIV = DIV_1

int main(void)
{
   unsigned char data;
   unsigned char str[]="HELLO";
   OpenUART1(UART_EN, UART_RX_ENABLE |
   UART_TX_ENABLE, 42);

   while(1){
      while(!DataRdyUART1());
      data = (char)ReadUART1();
      if(data=='a'){
         while(BusyUART1());
         putsUART1(str);
      }else{
         while(BusyUART1());
         putcUART1(data);
      }
   }
   return 0;
}
```

※"OpenUART1(UART_EN, UART_RX_ENABLE | UART_TX_ENABLE, 42);　"の部分は紙面の関係で2行で記載されていますが実際には1行で記述してください。

■PICD-32MX2ボードの設定

このプログラムを正しく動作させるため下記のようにPICD-32MX2ボードの各種設定をセットしてください。

ディップスイッチSW15のRS232C"A"系統のRXピンとTXピンをRF2とRF8に接続します。SW15のスイッチ1番と2番をONに設定し、その他のスイッチはOFFに設定してください。
その他、すべてのディップスイッチはOFF側に設定してください。

■コンパイルと書き込みの実行

これまでのチュートリアルと同様、ソースプログラムをビルドして、HEXファイルをPICD-32MX2のマイコンに書き込んでください。

■パソコン側の準備

このプログラムでは、RS232C通信を行います。パソコンのRS232CポートとPICD-32MX2の"A PORT"とかかれたポート(左側のコネクタ)を全結線ストレートケーブルで接続してください。

RS232Cのターミナルソフトは Windowsに標準で付属しているターミナルソフト"ハイパーターミナル"が使用できます。その他使い慣れているターミナルソフトがある場合にはそれをお使いください。

通信プロトコルは次のように設定してください。通信速度は115.2kbps、データ長8ビット、パリティなし、1ストップビット。

■動作確認

パソコン側から"a"以外の文字列を送信してください。送信した文字例がそのままエコーバックして返ってくることを確認します。続いて、文字"a"を送信してください。"HELLO"という文字列が返れば成功です。

■プログラムの解説

OpenUART1( );

UART1モジュールのモードを各種設定する関数です。

UART1を有効に設定し、送受信を有効にしています。

また通信速度の設定は下記の式で算出しています。

設定値=動作クロック÷(16×ボーレート)-1　左式に代入すると次のようになります。

80MHz÷(16×115.2kbps)-1=42.40･･･　よって小数点以下を切り捨てて42を指定します。

while(!DataRdyUART1());

DataRdyUART1( )関数はUART1モジュールで受信データがある場合には0、受信バッファにデータがある場合には、1を返します。よって、ここではデータを受信するまで待機します。

データがない場合には、while文で繰り返して受信待機します。関数の前の!は論理を反転させます。よって受信バッファにデータが到達すると値が0となりループから抜け出します。

data = (char)ReadUART1();

char型変数dataに受信したデータを代入します。

ReadUART1( )関数は受信データをバッファから取り出して、その内容を返します。(char)で明示的に、返すデータの型を指定しています。

if(data=='a'){

分岐命令で受信したデータを判定しています。C言語の場合、文字を表す場合にはシングルクォーテーション(')で囲みます。文字列の場合には、ダブルクォーテーション(")で囲みます。

while(BusyUART1());

BusyUART1( )関数は、UART1の送信ステータスを返す関数です。1でビジー状態、0でレディ状態を示します。ここでは送信が可能になるまでループを繰り返し待機します。

putsUART1(str);

putsUART1( )関数は文字列を送信する関数です。引数として文字列又は配列、ポインタを指定します。ここでは、unsigned char型の配列strを指定して"HELLO"という文字列を送信しています。

putcUART1(data);

putcUART1( )関数は、送信データを書き込む関数です。引数には変数を指定します。

チュートリアル⑥　～2行16文字キャラクタLCDを使用する～

PICD-32MXに搭載の2行16文字のオンボードキャラクタLCDに文字列を表示するプログラムを作成します。このLCDは、一般的に市販されている日立製コントローラHD44780コンパチブルです。

駆動に際しては、PICD-32MX2用のLCDライブラリをプロジェクトにインクルードして使用します。

ちょっとした変数内容の表示やLCD表示の実験用に使用できます。

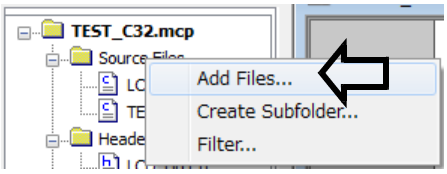
■プログラム作成前の準備

プログラム作成前にPICD-32MXのLCD駆動用ライブラリをプロジェクトに追加します。通常通り、プロジェクトを作成してください。
プロジェクトが作成できたら、左側のツリーウインドウの部分に下記の2つのファイルを追加します。

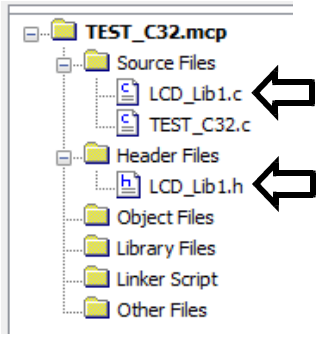
・LCD_Lib1.c
・LCD_Lib1.h

両方のファイルは、共に下記のディレクトリに入っています。
C:¥LV32MX Examples v1.00¥MMB

最初にツリーの一番上、"Sorce Files"を右クリックして、"Add Files"を選択します。ダイアログが表示されますので、上記ディレクトリ内にある"LCD_Lib1.c"を追加してください。



続いて、その下にある"Header Files"のフォルダを右クリックして同様に"Add Files"を選択します。ダイアログが表示されますので、上記ディレクトリ内にある"LCD_Lib1.h"を追加してください。
登録が完了すると、下図のように2つのファイルがプロジェクトに追加されます。



※TEST_C32.cファイルは、main関数を含むこれから作成するソースファイルです。

■プログラムの作成

```
#include "plib.h"

#pragma config FNOSC = PRIPLL
#pragma config POSCMOD = HS
#pragma config FPLLIDIV = DIV_2
#pragma config FPLLMUL = MUL_20
#pragma config FPBDIV = DIV_1
#pragma config FPLLODIV = DIV_1
#pragma config FWDTEN = OFF
#pragma config OSCIOFNC = ON

#define    LCD_DATA          LATB
#define    LCD_RS            LATBbits.LATB2
#define    LCD_E             LATBbits.LATB3

void  MCUInit(void);

int main(){
        MCUInit();
        char txt_a[] = "Hello";
        char txt_b[] = "World";
        lcd_init();
        lcd_clear();
        lcd_cmd(0x0C);
        lcd_cmd(0x80);
        lcd_str(txt_a);
        lcd_cmd(0xC0);
        lcd_str(txt_b);

        while(1){}
}
void MCUInit(){
        SYSTEMConfigPerformance(80000000);
        AD1PCFG = 0xFFFF;
        TRISB = 0;
        TRISC = 0;
        TRISD = 0;
        LATB = 0;
        LATC = 0;
        LATD = 0;
        delay_ms(100);
        return;
}
```

■PICD-32MX2の設定

使用するのは、2行×16文字のLCDのみですので、ディップスイッチSW20のスイッチ1番～6番をONに設定してください。その他のディップスイッチはすべてOFF側にセットします。

■動作確認

生成されたHEXファイルを、書き込んでください。LCDの1行目に"Hello"、2行目に"World"と表示されます。表示のコントラスト調整はP1ボリュームで行ってください。